Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

DIPLOMOVÁ PRÁCE

PLZEŇ, 2020 Bc. Jiří Vyskočil

University of West Bohemia
Faculty of Applied Sciences
Department of Cybernetics

MASTER'S THESIS

Scene type recognition of TV News broadcasts using visual data

Author:
Bc. Jiří Vyskočil

Supervisor:
Ing. Marek Hrúz, Ph.D.

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Aademický rok 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

Jméno a příjmení:	**Bc. Jiří VYSKOČIL**
Osobní číslo:	**A18N0042P**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Řídící a rozhodovací systémy**
Název tématu:	**Rozpoznávání typů scén zpravodajských pořadů z obrazových dat**
Vedoucí diplomové práce:	**Ing. Marek Hrúz, Ph.D.**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1. Navrhněte typy scén pro rozpoznávání.
2. Zvolte vhodnou topologii neuronové sítě pro účely rozpoznávání.
3. Natrénujte pomocí vhodných algoritmů danou neuronovou síť.
4. Experimentálně vyhodnoťte úspěšnost.
5. Diskutujte dosažené výsledky a navrhněte případné rozšíření.

Seznam doporučené literatury

Simonyan, Karen, and Andrew Zisserman. „Very deep convolutional networks for large-scale image recognition.“arXiv preprint arXiv:1409.1556 (2014).

Hochreiter, Sepp, and Jürgen Schmidhuber. „Long short-term memory.“ Neural computation9.8 (1997): 1735-1780. Chollet, François. „Keras.“ (2015).

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni. Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 1.5.2020

vlastnoruční podpis

Acknowledgement

Thank you to my supervisor, Ing. Marek Hrúz, Ph.D., for providing professional guidance and valuable advice while working on this thesis.

Poděkování

Děkuji vedoucímu diplomové práce Ing. Marku Hrúzovi, Ph.D. za odborné vedení a cenné rady při vypracování diplomové práce.

Abstract

Researchers of the Department of Cybernetics at the University of West Bohemia in Pilsen in cooperation with SpeechTech s.r.o. have developed a system, which automatically subtitles live broadcasts for Czech Television. The aim of this thesis is to extend the system for the "Události ČT" programme, with a scene recognizer using image data, appropriate sound filter aware of the scene type could be applied. Different neural network architectures were analyzed to develop a system capable of recognizing television news scenes. For evaluation of a network performance, a tool has been created, which generates an attention map, a model prediction including the correct class name for each input image and a confusion matrix.

By comparing an InceptionResNetV2 network to other backbone architectures, the results have shown, that the InceptionResNetV2 has the best performance during the learning phase. Thus, this network was further analyzed along with a compact MobileNetV2 network. The analyses explore, in addition to the different configurations of the models, the possibility of processing time-distributed image data. However, the testing phase has shown that the MobileNetV2 networks have more accurately classified the input images into correct classes, than the InceptionResNetV2 networks and that models, which process time-sequences of images, have lower recognition accuracy in most cases than networks, which perform classification based on a single input image. Besides these results, it can be unambiguously stated that the MobileNetV2 network is opening the possibility for practical usage, since it has considerably fewer parameters and the accuracy for classifying 9 classes was around 94 %, which is a very promising result.

Source files created for the purposes of this thesis are available on the website: `https://github.com/vyskocj/TV-News-Scene-Recognition`

Keywords: computer vision, digital image processing, artificial intelligence, scene recognition, TV News, neural networks, LSTM

Abstrakt

Výzkumnými pracovníky Katedry kybernetiky Západočeské univerzity v Plzni byl ve spolupráci s firmou SpeechTech s.r.o. vyvinut pro Českou televizi systém, který je schopen automaticky titulkovat přenosy z živého vysílání. S cílem rozvinout systém na pořad Události ČT vznikla tato diplomová práce, která se zabývá rozpoznáváním scén s použitím obrazových dat, aby následně dle typu scény mohl být aplikován příslušný zvukový filtr, který má schopnosti potlačení šumu pozadí a zvyšuje přesnost převodu řeči na text. Pro vývoj systému schopného rozpoznávat scény televizních událostí byly analyzovány různé architektury neuronových sítí. Pro vyhodnocení výkonu sítě byl vytvořen nástroj, který je schopen vygenerovat matici zmatení (confusion matrix) a pro každý vstupní obrázek mapu pozornosti (attention map) a predikci modelu včetně názvu třídy správné klasifikace.

Experiment porovnávající různé architektury neuronových sítí ukázal, že InceptionResNetV2 dosahuje nejlepších výsledků během učení v porovnání s ostatními sítěmi. Tudíž tahle síť byla následně analyzována společně s kompaktní architekturou MobileNetV2. Následné analýzy, kromě různých konfigurací sítí, prozkoumávaly i možnosti zpracování časově distribuovaných obrazových dat. Během testování se však ukázalo, že MobileNetV2 sítě jsou schopny přesněji klasifikovat než InceptionResNetV2 a že modely zpracovávající časové sekvence obrázků dosahují ve většině případů nižších přesností, než sítě, které provádí klasifikaci na základě jednoho vstupního obrazu. Z těchto výsledků lze jednoznačně konstatovat, že pro praktické využití je síť MobileNetV2 vhodnější i vzhledem k značně nižšímu celkovému počtu parametrů a s přesností klasifikace přibližně 94 %, což je příznivý výsledek.

Zdrojové soubory pro účely této práce jsou dostupné na stránkách: `https://github.com/vyskocj/TV-News-Scene-Recognition`

Klíčová slova: počítačové vidění, zpracování digitalizovaného obrazu, umělá inteligence, rozpoznávání scén, televizní zprávy, neuronové sítě, LSTM

Contents

Chapter 1

Introduction

In 2011, the SpeechTech s.r.o. in cooperation with the Department of the Cybernetics created an automatic subtitling system for Czech Television [41]. The system is designed for live sport and government session recordings. One of the possible research for the future is to extend speech recognition to a more complex task - such as television news broadcasts. There is a frequent environmental change in television news, which leads to different types of background noise. In extreme situations, such as live broadcasts with strong winds, it may be difficult to automatically recognize speech. A suitable filter must be used for these cases, but the same filter should not be used for low noise signals, e.g. where the anchorman is speaking from the studio. For this purpose, there is an idea to create a system that would be able to recognize different types of scenes using a computer vision technique. Consequently, multiple speech recognition filters could be used depending on the scene type to maximize the accuracy of automatic subtitling.

First, it is necessary to appropriately define the types of scenes in television news for recognition. One of the related goals is to find out whether it is possible to solve such a complex task using modern technologies, because it is not possible to define unambiguously all objects that each scene will always contain. Therefore, the use of standard computer vision methods might not lead to a correct result, so the basis of the model is the neural network. The first artificial neural network was introduced by Warren McCulloch and Walter Pitts in 1943 [37] and popular perceptron algorithm was introduced in 1957 by Frank Rosenblatt [45] as a simplified model of a biological network. One of the first greatest successes in computer vision was in the ImageNet Challenge 2012 using the Convolutional Neural Network (CNN) [48]. Since then, research has focused on developing artificial neural network architectures that are becoming state of the art in computer vision.

1.1 Motivation and objectives

The main goal of this thesis is to create a recognizer of television news scenes. Evaluation of the quality of the created models is based on the classification accuracy of the training and validation sets. But in addition to numerical evaluation, visual evaluation is also performed. For this purpose, a tool was created which automatically generates a file that can be opened by a web browser. This file contains outputs of the network for each input image and a heatmap showing the sections of the image the network is paying attention to. The best backbone architecture is eventually used to create a time distributed model. This model can make a full visual utility of the information from the video and suppress unwanted frames such as the transition between cuts. Finally, all relevant networks are tested on data from another television channel and an analysis, whether the time-series data processing brings benefits for recognition, is also performed.

In case of successful results of this thesis, research at the Department of Cybernetics can extend to several new directions. One practical example is the recognition of scenes on a television channel - such as parliament or sports broadcasting. This system could ensure automatic control of the speech recognition system, i.e. turning off the automatic subtitling if, for example, ads have started. This idea leads to the hypothesis of creating a network with a low number of parameters with satisfactory classification precision, i.e. worse accuracy than a large network, to be used online.

1.2 Thesis summary

This thesis has a following structure. Chapter 2 serves as a general introduction to neural networks, including activation functions, loss functions, and optimization. The relevant types of networks for purposes of this thesis are described in Chapter 3. In the following Chapter 4, scenes of TV News are defined and used datasets are characterized. A recognition system for practical application is designed in Chapter 5. In this chapter, the used neural network architectures for analyses are also described and a tool for evaluation of the network performance is explained. Finally, the experiments are presented and evaluated in Chapter 6.

Chapter 2

Artificial Neural Network

Neural networks are known for several decades and the popularity of this technology has grown in recent years into application in many research areas. Although it is a very powerful tool, its biggest drawback is the fact that it may require a large amount of computing power and vast collection of data to train complex systems.

The basic idea was to create a model of a biological neural network which consists of neurons that are connected by so-called synapses. To explain simply, this network can be recognized as a combination of neurons. Each neuron has an activation function and several inputs that are weighted. A neuron becomes active if the sum of the weighted inputs is higher than the threshold of the neuron. One of the basic networks is a perceptron [45, 46] one, which is defined as a single-layer neural network with a simple activation:

$$y_l = \left\{ \begin{array}{ll} 1: & \sum_{n=1}^{N} w_{l,n} \cdot x_n + b_l > 0 \\ 0: & otherwise \end{array} \right. \tag{2.1}$$

where $w_{l,n}$ are weights, x_n are inputs, b_l is bias and y_l is output of l-th neuron. Convergence is assured, if the learning set is linearly separable. In general, linear separability cannot always be ensured and this is also the reason for increasing the depth of the network. If we now consider a two-layer network (Figure 2.1), the output can be obtained according to the following formula:

$$y_l = g(\sum_{m=1}^{M} [v_{l,m} \cdot f(\sum_{n=1}^{N} w_{m,n} \cdot x_n + b_m)] + d_l) \tag{2.2}$$

where f and g are activation functions for first layer and second layer, $v_{l,m}$ are weights of second layer, d_l are biases of second layer. Generally, these layers are called dense layers. Such part of the network is also described as fully-connected layers in models with a more complex structure, consisting of different layer types.

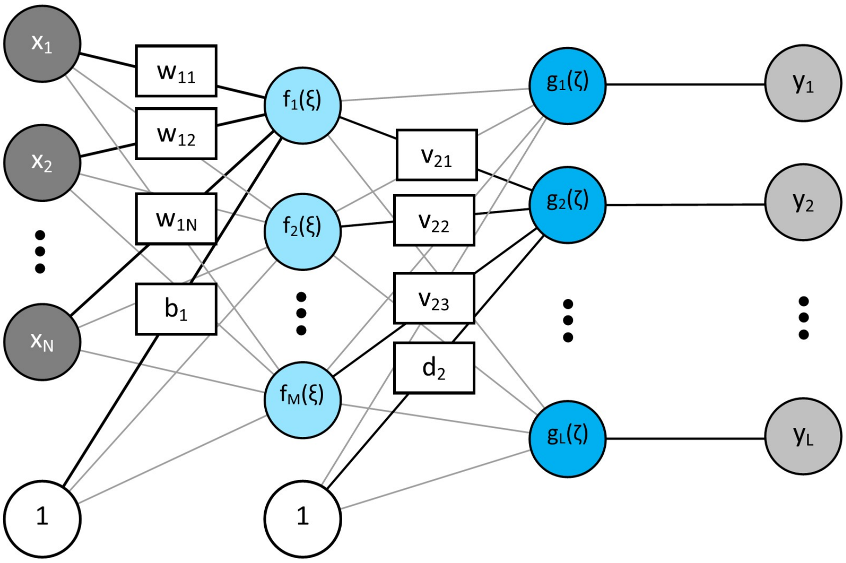


Figure 2.1: Example of a two-layer neural network.

2.1 Activation functions

One important aspect of neural network design is topology. To get a network with a good performance, it is also important to choose the activation functions that affect the selection of information in each layer in the multidimensional space. From a cybernetics point of view, each neuron controls the output of previous neurons to obtain the desired output from the system. By using the non-linear activation functions, a neural network can approximate any function with sufficient accuracy [25].

2.1.1 Logistic function

The logistic function is also known as the sigmoid function, mainly due to its S-shape characteristic. This function was introduced by Pierre François Verhulst between 1838 and 1847 in three papers [8]. It is a standard function used in neural networks that maps values from the real values to the probability space. The function is defined according to the following formula [8]:

$$f(\xi) = \frac{e^{\xi}}{1+e^{\xi}} = \frac{1}{1+e^{-\xi}} \tag{2.3}$$

The function is differentiable and monotonic - but the derivative is not monotonic. The negative aspect of this function is that learning process may come to a halt during the training time [14]. It was one of the motivations for exploring new functions.

2.1.2 Rectified linear unit

In 2000, Hahnloser first introduced the most popular activation function for state of the art networks that is known as ReLU [20, 16, 42, 30]. One of the main advantages of this function is that the network can learn several times faster than a network using sigmoid functions [30], because gradients are able to flow when the input to the function is positive [42]. This simple and effective function is defined as [42, 20]:

$$f(\xi) = max(0, \xi) \tag{2.4}$$

According to the formula, a problem might occur, because the function is not differentiable at zero. This is solved by modifications of this function - such as Softplus [15], ELU [7], or Swish [43]. Other well known modifications include Leaky ReLU, Parametric ReLU [21], or maxout [17].

2.1.3 Softmax function

With neural networks designed for classification, there is often a need to return a probability vector. The individual elements of the vector represent the probability with which the input signal belongs to the output class. Ideally, the output should be a one-hot vector, i.e. the input signal is classified with the probability of 1 into one class and the same signal belongs to the other classes with the probability of 0. Since the output values from the previous layers can be distributed in the range of real numbers, the last layer - also known as the classification layer - uses the softmax function defined as [16]:

$$\sigma(\xi)_j = \frac{e^{\xi_j}}{\sum_{k=1}^{K} e^{\xi_k}} \tag{2.5}$$

where $\sigma(\xi)_j$ is the probability of the input belonging to j-th class.

2.2 Loss function

It is necessary to define an optimization target before learning starts. In our case, it is a minimization of the criteria function, which is represented by a classification error [16]. A very popular loss function is the mean square error that is not suitable for classification. This criterion is mainly used for regression problems, such as segmentation or key-points detection. Since probabilities given by softmax function can be expected at the output, it is simple to utilize the principle of maximum likelihood [16] - it means that categorical cross-entropy shall be used between the training data and the model's predictions as the loss function [16]:

$$L(p, q) = -\sum_x p(x) \cdot \log q(x) \tag{2.6}$$

where p is information provided by supervisor - i.e. expected output - and q is model output. The sum is calculated for all inputs x. Ideally, the predictions shall be equal to the expected output. In this case, the network output is represented by a one-hot vector and the loss is equal to zero [4].

2.3 Optimization

When a neural network model is created, an optimization has to be performed. Like a human, the machine is unable to understand the problem immediately. Optimization is a method to change the parameters of the network to reduce losses. The parameters can be pre-trained or initialized randomly. Choosing a pre-trained network for a similar task can bring many benefits [53]. This network can learn faster than a network with randomly initialized parameters. The reason is that such a network can extract significant pieces of information from the input. Training a model that has already been optimized is called transfer learning.

The machine learning can be divided into several categories: *supervised learning*, *unsupervised learning*, and *reinforcement learning*. For *supervised learning*, a pair of inputs and required outputs are used during fitting of the model. After successful training, the network can produce outputs corresponding to the training data. In *unsupervised learning*, the model looks for continuity in data. This learning is also called clustering. In *reinforcement learning*, the model performs actions, that are rewarded by the environment. The main idea of this learning is to find a balance between exploration (uncharted area) and exploitation (current knowledge) [60]. In case of this thesis, a pair of inputs and required outputs are available, therefore, a feedforward neural network can be used for the given topic with the backpropagation algorithm to train it.

The backpropagation was introduced in 1986 [47] to train a neural network with one or two layers [16]. The main idea is to compare the prediction with the desired output. The prediction error is propagated back to the input as the gradient of the loss function $E(X, \theta)$ is calculated [2]. This value is used to update the neuron parameters θ, i.e. weights and bias [2]:

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial E(X, \theta_t)}{\partial \theta} \tag{2.7}$$

where X is set of input-output pairs, α is the learning rate, and t is the iteration. This technique is also called gradient descent. Non-trainable parameters which are defined before the learning are called hyperparameters. Backpropagation has become so popular that it had a vital role in discovering deep neural networks for image and speech recognition [2].

2.3.1 Stochastic Gradient Descent

The main problem of gradient descent is that the partial derivative needs to be computed for a single parameter. With a deep neural network and with a large dataset, there is a huge number of derivations to compute. In fact, a large dataset is necessary for good generalization but at the cost of more computationally expensive learning [16]. This can be solved by introducing stochastics in learning.

The idea of stochastic gradient descent is a calculation of estimated gradients. The estimation is calculated in each step by using minibatch - i.e. small set of samples [16]. The minibatch samples X_b are chosen uniformly from the training set X. The size of a minibatch B can be defined before fitting the network in range of tens to a few hundred [16]. While maintaining the definitions of the gradient descent formula 2.7, it is possible to iteratively calculate the parameters for the stochastic variant [16]:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{B} \cdot \nabla_\theta \sum_b E(X_b, \theta_t) \tag{2.8}$$

With this modification, deep neural networks can be trained in shorter time, compared to the deterministic variant. In general, the use of gradient descent does not guarantee that the global minimum of the loss function is achieved during the learning. The learning rate α is the only parameter that can ensure a local minimum. Learning can be a slow process for some tasks with this optimizer. Therefore, adjustments have been introduced to allow the use of a momentum.

The momentum is a powerful method that is designed to speed up learning. The idea is to accumulate the average of past gradients and move in their direction [16]. The average of past gradients represents a path that has already been traveled. This allows getting over local extremes while the network is learning. There is an analogy from physics, the Newton's laws of motion: the algorithm includes a velocity variable v and a hyperparameter $\gamma \in [0, 1)$ indicating how quickly the contributions of the previous gradients exponentially decay [16]:

$$\begin{aligned} v_{t+1} &= \gamma \cdot v_t - \frac{\alpha}{B} \cdot \nabla_\theta \sum_b E(X_b, \theta_t) \\ \theta_{t+1} &= \theta_t + v_{t+1} \end{aligned} \tag{2.9}$$

Another variant of momentum is a Nesterov momentum, which is evaluated after the current velocity is applied [16]. But there is one more problem with the gradient. The gradient is defined as a direction and rate of the fastest increase. The main goal of learning is to minimize the loss function, i.e. ideally to find a global minimum. The network is learning slowly when the gradient is small. In learning, the optimizer can skip the local minimum or the optimizer can get stuck when the gradient is large. For this purpose, algorithms with adaptive learning rates were introduced, such as AdaGrad [10], RMSprop [59], Adam [28].

2.3.2 RMSprop optimizer

One modification of stochastic gradient descent is AdaGrad that adapts the learning rate individually for each parameter. Parameters with a large gradient of the loss function have a relatively large decrease in their learning rate and parameters with a small gradient have a corresponding increase in their learning rate [16]. RMSprop follows the AdaGrad principle by solving the problem with accumulated gradients. This accumulation is changed into an exponentially weighted moving average - the length scale of the moving average controls the hyperparameter ρ [16]:

$$\begin{aligned} g_{t+1} &= \frac{1}{B} \cdot \nabla_\theta \sum_b E(X_b, \theta_t) \\ r_{t+1} &= \rho \cdot r_t + (1-\rho) \cdot g_{t+1}^2 \\ \theta_{t+1} &= \theta_t - \frac{\alpha}{\sqrt{r_{t+1}} + \delta} \cdot g_{t+1} \end{aligned} \tag{2.10}$$

where δ is a constant for numerical stability (this constant is chosen to be as small as numerically viable). With this modification, further history information is suppressed, which increases the efficiency of deep neural network learning [16]. By removing all occurrences of the constant ρ, the equations corresponding to the AdaGrad algorithm are obtained.

2.3.3 Adam optimizer

Adam is an effective adaptive optimization algorithm that realizes the benefits of RMSprop and momentum. Similarly to RMSprop, Adam uses an estimate of the first-order moment of the gradient [16]. Besides, the second-order moment of the gradient is also estimated. Corresponding to Keras API [6], Adam introduces hyperparameters β_1 and β_2, which have the same function as ρ in RMSprop: to control the decay rates of the estimations. Adam also includes bias corrections, i.e. $\hat{s}$ and $\hat{r}$, to the estimates of both moments [16, 28]:

$$\begin{aligned}
g_{t+1} &= \frac{1}{B} \cdot \nabla_\theta \sum_b E(X_b, \theta_t) \\
s_{t+1} &= \beta_1 \cdot s_t + (1 - \beta_1) \cdot g_{t+1} \\
r_{t+1} &= \beta_2 \cdot r_t + (1 - \beta_2) \cdot g_{t+1}^2 \\
\hat{s}_{t+1} &= \frac{s_{t+1}}{1 - \beta_1} \\
\hat{r}_{t+1} &= \frac{r_{t+1}}{1 - \beta_2} \\
\theta_{t+1} &= \theta_t - \frac{\alpha \cdot \hat{s}_{t+1}}{\sqrt{\hat{r}_{t+1}} + \delta}
\end{aligned} \tag{2.11}$$

Adam is robust to the choice of hyperparameters [16] and due to its efficiency, there are some variants, e.g. Adamax [28], Nadam [9] or AMSGrad [44]. According to the article [44], AMSGrad uses a maximum of all r_t, i.e. biased second-order moment estimate:

$$\begin{aligned}
\hat{s}_{t+1} &= s_{t+1} \\
\hat{r}_{t+1} &= max(\hat{r}_t, r_{t+1})
\end{aligned} \tag{2.12}$$

This simple modification provides the algorithm with long-term memory of past gradients. In some cases, it actually shows improvements [44].

Chapter 3

Types of Neural Networks

To use the neural networks in certain research or complex tasks, new types of this technology had to be evolved. Many innovations around neural networks are based on biological counterparts - such as long-short term memory or convolutional networks. This is one of the reasons why the neural networks can be considered as a suitable model for computer vision or speech recognition.

The neural network model described in the Chapter 2 could be used to some extent for computer vision, but such model would not be ideal for image classification. There is some correlation between nearby image elements that would not be adequately reflected by using fully-connected layers. Creating such model would lead to many inconveniences. This model would contain a large number of parameters even for a relatively small picture. Therefore, learning would also be very difficult. After optimizing, this model will tend to classify objects according to the positions that were most often encountered during training. These facts have led to the development of a convolutional neural network that suppresses these disadvantages.

3.1 Convolutional Neural Network

Convolution is a fundamental mathematical operation for control theory. It is used in cases where an expected or investigated signal is a combination of two different functions. One of those functions is usually input signal and the other is a model, for example, a filter or a regulator. This operation is defined as integral of the two functions f and g, where one of them is reserved and shifted [16]:

$$f(t) * g(t) = \int_{\mathbb{R}} f(\tau) \cdot g(t-\tau) d\tau \tag{3.1}$$

The equation can only be used for one-dimensional signals. For image processing this formula (in discrete form) can be applied in the histogram smoothing, where t represents brightness value. Generally, an image can be considered a two-dimensional matrix that has its width, length, and individual elements of the matrix indicate brightness. From a practical point of view, the image can also be considered as three-dimensional matrix, where the third dimension is a color model, most often RGB. Convolution for image processing can be defined as [16]:

$$(I * K)[i, j] = \sum_{m} \sum_{n} I[m, n] \cdot K[i - m, j - n] \tag{3.2}$$

where I is image and K is kernel, which can also be called a filter depending on the application. Before utilizing neural networks in computer vision, the convolution had its application that it became a standard operation for digitized image processing. Basic applications include image blurring and edge detectors. As mentioned earlier, the surrounding points of each pixel are related. With the right filter, it is possible to extract various information from the images - either edges or color composition. Those filters can be subsequently combined, which leads to object recognition in the images. Therefore, convolutional neural networks were introduced.

The inspiration came from biological processes [36] - the neurons' connectivity of the convolution layer is similar to the visual cortex organization [36, 11]. According to the Deep Learning book Chapter 9.2 [16], in the following lines, the convolution brings three important ideas to neural networks: *sparse interactions*, *parameter sharing*, and *equivariant representations*. Although the kernel is smaller than the input image, the model can detect small, meaningful features. It results in storing a few parameters, thereby *sparse interactions* are accomplished. Whereas in a traditional neural network, each parameter of the neuron is used once while computing an output, in the convolutional network, *parameters are shared* to using them for more than one function in a model. Convolution produces a two-dimensional map of features that each depends on the input. *Equivariant representations* means that with moving objects in the input its representation will move the same amount in the output.

On the contrary, the convolutional network is not naturally equivariant to some transformations, such as rotation [16]. It may be demanded for some reason, e.g. recognition of buildings from the street perspective. If any transformation, that is missing in training data, would be useful, image augmentation should be performed.

This type of network consists of several kinds of layers, each of which has its own importance. Individual layers are stacked where convolutional layers and pooling layers represent the function of image processing, i.e. extracting important information from the image. Behind the last layer of the convolutional part are so-called top layers that regularly represent a traditional neural network. Such a basic architecture can be used for classification or regression.

3.1.1 Convolutional layer

As the name suggests, the base of this layer is the convolution described in Chapter 3.1, equation 3.2. Except for the activation function described in the Chapter 2.1, this layer defines parameters such as: *size* and *number* of kernels, *strides*, and *padding*. These parameters depend on the size of the field of view, which is also called the receptive field. With deeper neural network, the receptive field grows larger. The output of each layer is called a feature map. The receptive field refers to the part of the image that is affected by one point of the feature map. Figure 3.1 shows three consecutive convolutional layers where the dark blue area indicates the field affected by one kernel of the second layer and the light blue area marks the receptive field of one point in layer 3 [31].

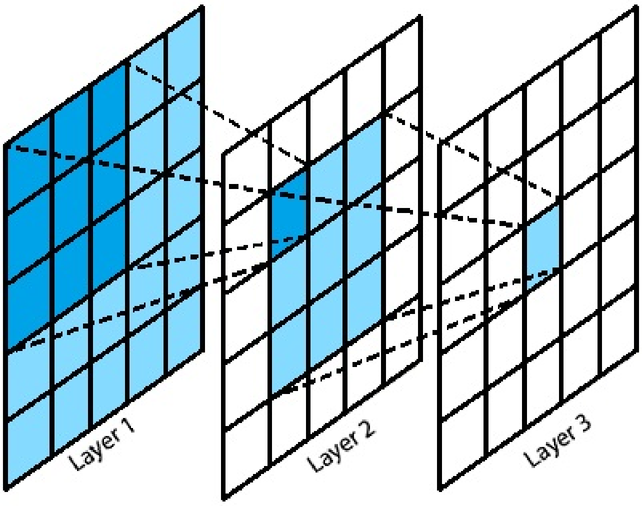


Figure 3.1: The receptive field of each convolution layer with a 3 × 3 kernel. Obtained from [31], modified.

Kernel *size* is defined as width × height × channel. The channel depends on the color model that represents the number of color scales of the input image. For grey-scale images, the channel is equal to 1 and kernels can be set to extract intensity of sections in the image. The channel of RGB images is equal to 3 and the network can extract different color combinations from the image. The width and height of the kernel are in pixels, a 3 × 3 kernel size is often chosen. One of the reasons is a lower number of parameters even in the case of the same receptive field but a deeper network. For example, when extraction of information from a 5 × 5 sectors of an image is required, the number of weights is 25. By using

3 × 3 kernels in two consecutive convolutional layers - see Figure 3.1 - there is 9 + 9 number of weights.

As in the case of the Haar-like feature [40], in convolutional networks multiple various filters are necessary to describe an object. Although the network learns to set the filters itself, determining the *number* of kernels affects the the networks' performance. As a rule, the number of kernels in a layer increases with the depth of the network. The feature map, i.e. the output of the convolution layer, is obtained by moving the kernel by the specified number of steps that the *strides* parameter specifies. With larger steps, the feature map gets smaller in width and height. The last basic parameter is *padding*, which is used for situations, where the same output dimensions of each layer are required. This is especially demanded for residual networks.

3.1.2 Pooling layer

The sequential stacking of convolution layers produces a large number of features describing individual parts of the input image. To keep the most important features, it is advisable to use a layer that expresses the most worthy information. For this purpose, pooling layers, that summarize data at the end of a certain block of convolutional layers, were introduced. Moreover, the use of pooling layers results in a representation of approximately invariant to small translations [16]. As a reminder, the convolutional neural network adjusts the weight parameters to detect edges, corners, or color composition. The response to the input of such kernels is highest when, for example, the corner detector has found a corresponding corner in the image. This is the main reason why the most viable data can be found by extracting the highest values of the feature map within a rectangular neighborhood - this is performed by using the max-pooling layer. This layer has two basic parameters: the *size* of the pooling window and *strides* representing steps of moving the window.

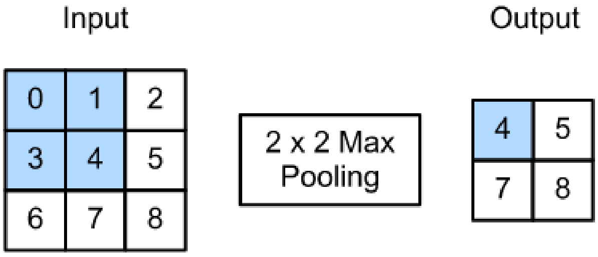


Figure 3.2: A maximum pooling layer with a 2 × 2 window shape. The image was obtained from [64].

Another variant is average pooling, that computes the average of the pooling window. Behind all convolutional and pooling layers, there can be inserted a Flatten layer, which reshapes the feature map to a vector, or a global pooling layer. Using the Global Average Pooling, the network takes the average of each feature map, and the resulting vector is fed directly into the top layer [32]. This can bring improvements to network classification accuracy over the use of the Flatten layer.

3.2 Residual Network

Convolutional neural networks adjust the filters during training at different depths [63]. However, as the depth of the network increases, learning becomes more difficult [21, 22]. With stacking layers, the susceptibility to vanishing or exploding gradients used to update parameters increases [22, 14]. The network is unable to set the parameters correctly, and this prevents convergence [22]. This problem can be largely avoided by adding normalization layers [22, 16]. To make the neural networks deeper, there is a solution by using skip connections to jump over some layers. This is illustrated in Figure 3.3 in which the left part of the image shows a regular block and the right half of the image displays a residual block. Using a residual layer, the network learns residual mapping instead of underlying mapping, as in the case of a regular convolution network [22].

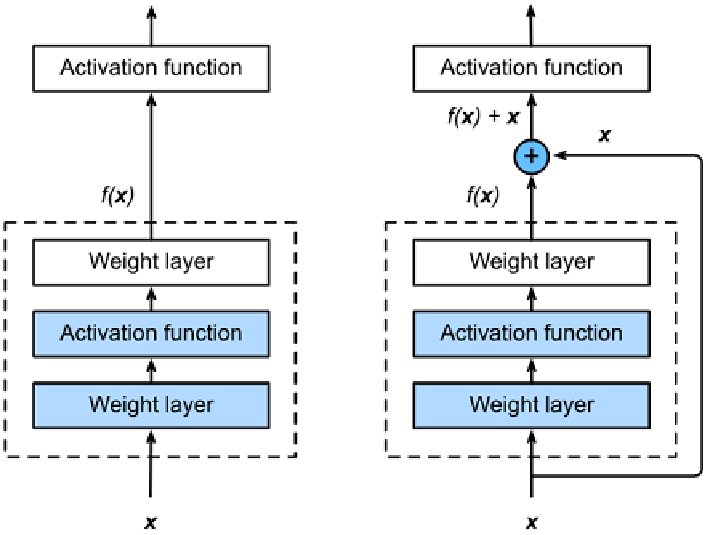


Figure 3.3: Comparison of a regular block (on the left) and a residual block (on the right). The image was obtained from [64].

While standard convolutional networks reach depths of up to 20 layers, the residual neural networks can reach several times more. In article [22] there were introduced ResNet networks with 152 layers that are 8 times deeper than VGG networks and still having lower complexity. They have won the first place on the ILSVRC 2015 classification task [49] and they also presented analysis with 1000 layers. Neural networks of this depth were not trainable before the introduction of the skip connection inside the model [22].

3.3 Recurrent Neural Network

Although neural networks are a powerful tool, the feedforward variant lacks state information which, if available, can provide benefits in some situations. The decision is affected by the previous state, whose history can be represented by neurons with recurrent connections [38]. This enables dynamic behavior that is able to respond to a sequence of inputs. The state is time-dependent in many applications, and since the neural network is a model of the biological brain, the state is called memory.

Theoretically, recurrent neural networks should be able to model long-term dependencies. From a practical point of view, there is a deep network structure due to the length of the input sequence. This involves the problem of vanishing gradients which has a strong impact on learning convergence. Therefore, new variants have been explored to help improve the stability of gradients. The first recurrent network was the Hopfield network [24] but the greatest advancement was the introduction of an LSTM network [23], that is able to vanquish with the problems of vanishing gradients.

3.3.1 Long short-term memory

The challenge to learn the long-term dependencies effectively was reached by introducing the LSTM network [23]. The core of this network is a memory cell consisting of several gates - i.e. input gate, output gate, and forget gate. The gates represent the function of a regulator that controls the information flow in the memory at each step. The state of memory cell changes according to the recent input events through time. The memory cell structure is shown in Figure 3.4 which is taken from [64].

According to [64] and [23] in the following lines, the input information is concatenated with a hidden state. The hidden state is used to pass information, which is affected by the output gate, to the next state. The output gate determines when the memory information of the cell is used for other processing. This gate can pass through all memory information, although it can retain all memory information only within the memory cell. The forget gate decides what memory information it keeps and discards. The input gate controls how much cell takes

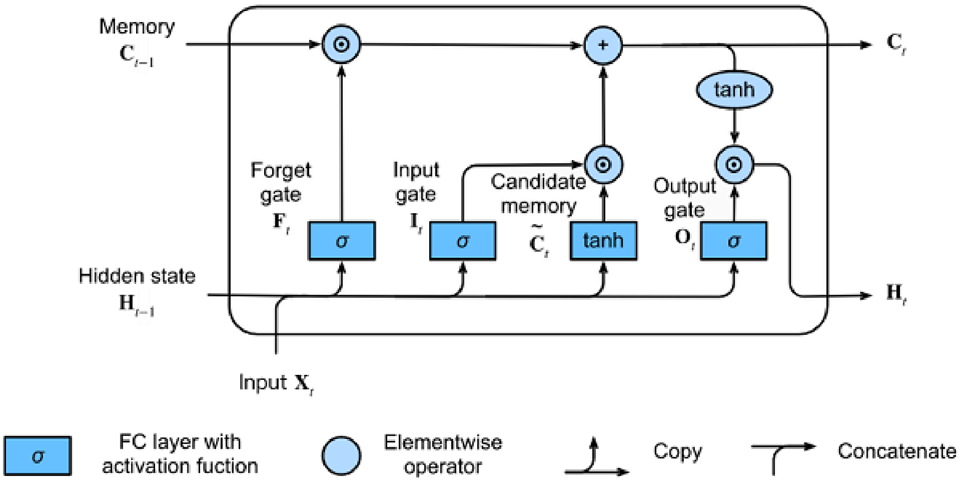


Figure 3.4: Memory cell of an LSTM network. The image was obtained from [64].

new information via candidate memory which is modified based on a sigmoidal function.

Compared to its predecessors, the LSTM network has been proven to be so robust that it has found applications in many specializations. This technology is the most utilized in applications with speech recognition [18, 50], grammar learning [13] or semantic parsing [27]. However, it also turned out that the network is able to predict the time-series of data points [12]. In computer vision, the network has applications in object co-segmentation [62], handwritting recognition [19] and sign language recognition [33].

Chapter 4

Data

For supervised learning, an input-output pair is required that contains input data and desired outputs. The provided dataset highly influences the quality of the model. Individual classes should be well distributed and contain as much data as possible. The dataset is often balanced because a network is then more motivated to learn the characteristics of each class. However, if there is no balanced data, augmentation can be performed. Besides, in some cases, different class sizes may be desired. Such a network is provided with information about the class, which it can expect more often on the input.

4.1 Scenes definition

The main idea of this thesis is to recognize several television news scenes of the Czech TV with a possible extension for additional TV channels. These scenes are divided into nine following classes:

- **Graphics**: a scene with graphics adjustments, e.g. photo including text message. Or scene without a real environment and characters, such as opening titles, TV News transitions, animations, statistics, etc.
- **Historic**: a scene containing old videos and pictures that are typically black and white.
- **Indoor**: a scene inside a building or a hall that are not inside the studio
- **Studio**: a scene inside a TV news studio with a anchorman
- **Mix**: a split-screen scene containing two or more different views. For example, conversations between speakers at two remote locations.
- **Outdoor country**: a scene containing a natural environment. There are no roads and/or buildings in the main field of view.

- **Outdoor human-made**: a scene containing human-made structures, such as buildings, roads, or highways.
- **Speech**: a scene of a press conference speakers.
- **Other**: a scene that is not specified. It may include situations in which it is not possible to classify as another class. For example, a close-up shot of stocked materials.

Compared to object classification, there are no trivial features that accurately represent each type of scene. The scene classification classes are very extensive, for example, in the outdoor country, a lot of greenery is expected, such as forest nature or hills. Something similar can be found in a city, though, e.g. in parks. In object classification, the scenes should not be too complex to train an high-quality network. A complex scene is an image containing too many objects that can be overlapped, or the image containing visual distortion. This makes the learning process more difficult when it is harder for the neural network to find the right characteristics of objects in the scene.

4.2 Created datasets

The model is optimized for a training dataset and evaluated on a validation dataset after each epoch. One epoch, i.e. training cycle, occurs when all data from the training set were used exactly once for updating parameters. These sets usually contain data of similar density. The validation set is used to avoid overfitting, because such a model is no longer able to generalize. The validation dataset is also used for tuning the hyperparameters. The last dataset is a testing set that is independent of the previous datasets. It is used for the final evaluation of the model performance.

Datasets for neural network optimization are obtained from the Czech television broadcasts. These data were divided into sequential and isolated images. In the same way, a test dataset was created from the broadcast of TV Prima. Size of the isolated images is $180 \times 320 \times 3$ (height $\times$ width $\times$ channel). The sequential set for time-distributed neural networks is extended by 25 frames. The frames of the sequential set were taken every 0.2 seconds apart. The amount of images (or sequences of images in case of time-distributed models) is summarized in Table 4.1 for each dataset. In the Figure 4.1, examples of the Czech television broadcasts dataset are displayed.

	Dataset		
Used for:	Training	Validation	Test
CNN	27183	3028	8300
CNN-LSTM	5392	1040	332

Table 4.1: The number of inputs in created datasets.

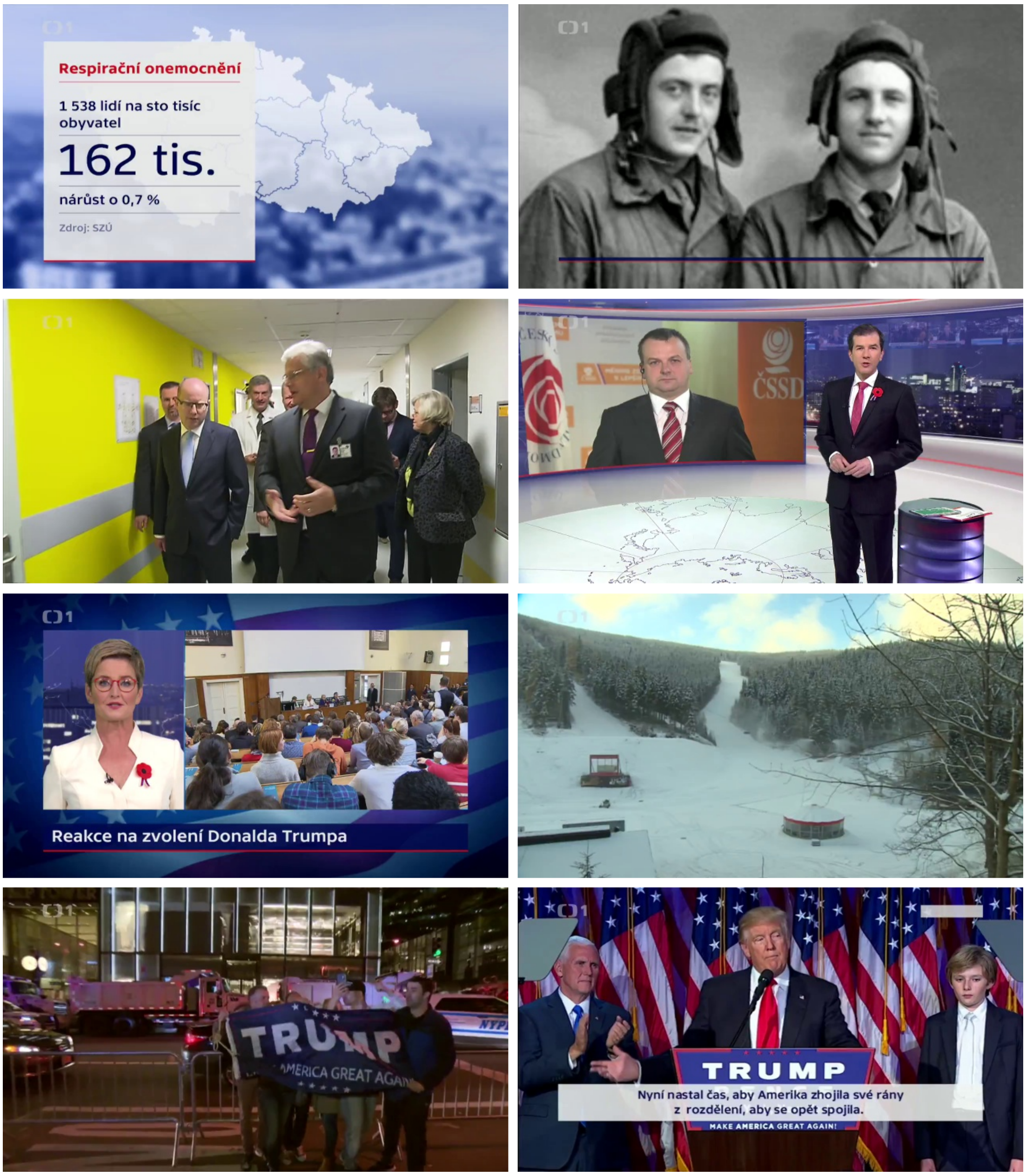

Respirační onemocnění
1 538 lidí na sto tisíc obyvatel
162 tis.
nárůst o 0,7 %
Zdroj: SZÚ
ČSSD
Reakce na zvolení Donalda Trumpa
TRUMP
AMERICA GREAT AGAIN
TRUMP
Nyní nastal čas, aby Amerika zhojila své rány z rozdělení, aby se opět spojila.
MAKE AMERICA GREAT AGAIN!

Figure 4.1: Examples of classes from Czech TV data: Graphics, History, Indoor, Studio, Mix, Outdoor country, Outdoor human-made, Speech. An extended form of examples is available in the Appendix A

Chapter 5

Implementation

Python 3 was chosen as the main programming language for elaboration of this thesis. The advantage of using this language is high readability of scripts and the ability to easily download missing libraries via the command line. Outcome application includes the following basic functionality:

- Obtaining image data from a video track.
- Creating dataset from image data.
- Designing and optimizing the required models.
- Generating the model evaluation:
  - A confusion matrix.
  - An HTML file with input data and model's predictions.
  - An HTML file that compares attention maps of various models.

A system capable of recognizing scenes from isolated images was created at an early stage of development. These models were acceptable, but the system was extended to the processing of time-distributed data to reduce the classification error. The design of the time-distributed recognition system is shown in Figure 5.1 that consists of sequential input data, cut detector, neural network, and classification to the defined scenes.

5.1 Cut Detector

First, it is necessary to extract relevant data from the video track. In the case of a time-independent model, the data are isolated images. Consecution of these images was not taken into consideration. Thus, a sequence of images can be used for scene recognition instead. One of the tasks of this thesis is to figure out how to divide input data in a way, that it does not contain more than one scene.

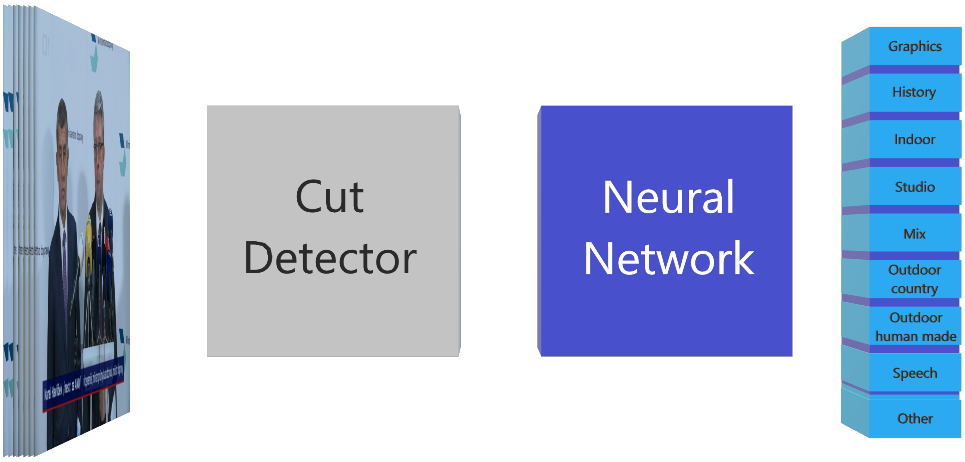


Figure 5.1: Architectural design of a recognition system.

The selected solution is introducing Cut Detector into the system located between the broadcast video and the neural network. Cut Detector provides input data from a single scene to the network. According to the paper [35] in the following lines, this algorithm utilizes two phases of cut detection: a *scoring phase*, and a *decision phase*. In the *scoring phase*, a similarity score is given to each pair of two consecutive video frames. The score can be calculated by the sum of absolute differences, or by the difference between the two histograms. In the *decision phase*, all previously calculated scores are evaluated, then a video cut is detected if the score is higher than given threshold.

The reimplementation was realized by Ing. Pavel Campr, Ph.D. and provided by the supervisor of this thesis, Ing. Marek Hrúz, Ph.D. The script and the configuration file can be found in a separate folder from the source scripts created by the author of this thesis.

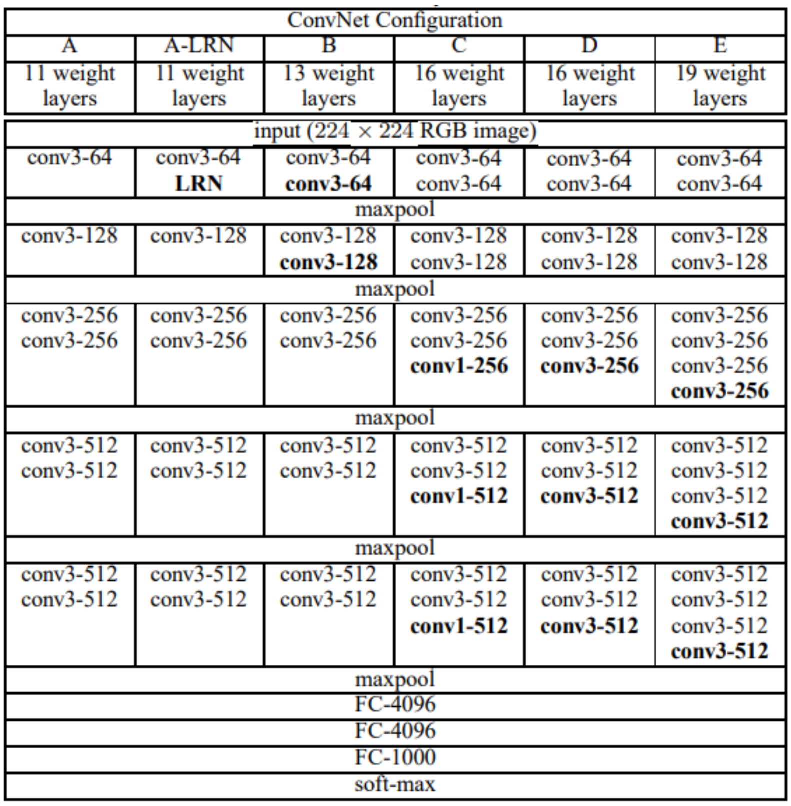
5.2 Used Neural Network architectures

The Keras library [6] was used for neural network implementation. It is a high-level API that utilizes TensorFlow [1], CNTK [51], and Theano [58] backends. For purposes of this thesis, the TensorFlow backend was chosen. The main feature of this framework is an easy model building and portability of the model between different platforms [1]. NVIDIA CUDA® Deep Neural Network library [5] was also used to provide optimization of standard routines such as convolution, pooling, normalization, and activation layers.

In addition to creating the neural network model fast and easy, the Keras library also provides optimization algorithms (see Chapter 2.3) and pre-trained networks. These networks are stored at the Application module that includes architectures for image classification with weights trained on ImageNet. The ImageNet [49] is an image database contains millions of images according to the WordNet [39] hierarchy. This database aims to illustrate each synonym set [49], i.e. word phrases or multiple word descriptions. The used pre-trained models for the elaboration of this thesis are described in the following subsections.

5.2.1 Visual Geometry Group

Simonyan and Zisserman from the University of Oxford have presented one of the most famous neural network models in the paper [54]. VGG networks have a very simple structure that consists of 3 × 3 convolutional layers and 2 × 2 max-pooling layers. Behind all of the convolution layers there are three fully connected layers - the first two have 4096 neurons and the last layer performs classification by softmax function to the 1000 classes. The network configurations are shown in figure 5.2. These configurations differ only in their depth. The most known are the D and E configurations having 16 and 19 weight layers. The difference in C and D configuration is only in the size of the receptive field at the end of the last three convolutional blocks.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 5.2: Configuration of VGG models, convolutional layers are denoted as *conv<receptive field size>-<number of channels>*. Captured from [54].

This network is still capable of impressive performance. It is also utilized in several applications, such as object detection and segmentation. For example, the VGG16 becomes the backbone architecture of the Single Shot Detector [34] and SegNet [3].

5.2.2 Inception networks

In the article [56] an efficient neural network architecture for computer vision, that enhances the strength of Network in Network [32], was proposed. The architecture contains inception blocks that are visualized in figure 5.3. These blocks consist of parallel paths in which the convolution layers and the pooling layers are sequentially stacked. In each path various sizes of convolutional filters are used to extract worthy information from the input. 1 × 1 filters are puts in front of the 5 × 5 and 3 × 3 convolutions to reduce complexity of the network. Finally, the outputs of each path are concatenated to provide the output from the inception block.

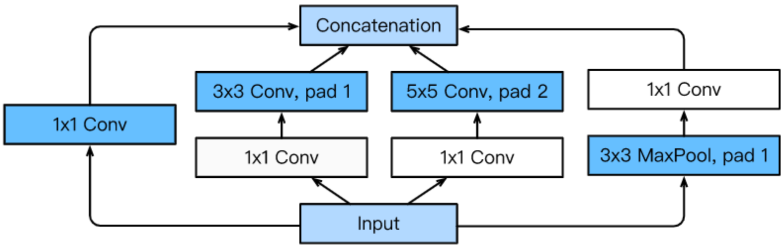


Figure 5.3: Structure of an inception block. Obtained from [64]

There are several variants of inception networks. For example, InceptionV3 [57] replaces 5 × 5 and 3 × 3 filters by mini-networks. The n × n windows are factorized to the combination of 1 × n and n × 1 filters. Another variant is InceptionResNet [55], which introduces residual connection into the network - see Chapter 3.2. It allows making the network deeper for better recognition performance.

5.2.3 Mobile networks

The presented architectures above may achieve excellent results, but such models may also be computationally demanding. The motivation of MobileNets [26] was to reduce the trainable parameters to maintain the recognition efficiency. The presented networks are primarily intended for mobile devices and embedded vision applications.

The article [26] will be elaborated upon in the following lines. The main idea behind these networks is the usage of the Depthwise Separable Convolution. Standard convolution is factorized into a depthwise convolution and a pointwise convolution. First, the depthwise convolution applies a single filter to each input channel. Then, the pointwise convolution applies a 1 × 1 convolution to combine the depthwise convolution outputs. This divides the standard convolution into two layers, the depthwise convolution for filtering and the pointwise convolution for combining. This factorization drastically reduces the model size and complexity.

5.3 Network performance evaluation

When the training ends, the Keras framework [6] provides a history of loss function and accuracy in each epoch. These values give information about the training progress, e.g. whether overfitting has occurred. However, these functions do not evaluate which classes of the network are classified incorrectly. A confusion matrix is usually used for this description of the network performance. It is a chart, which contains the number of correct and incorrect predictions that are broken down by each class. Each value of this matrix provides information about how many times the input belonging to the i-th class has been classified into the j-th class. For each fitting of the model, the confusion matrix is automatically generated in the output directory.

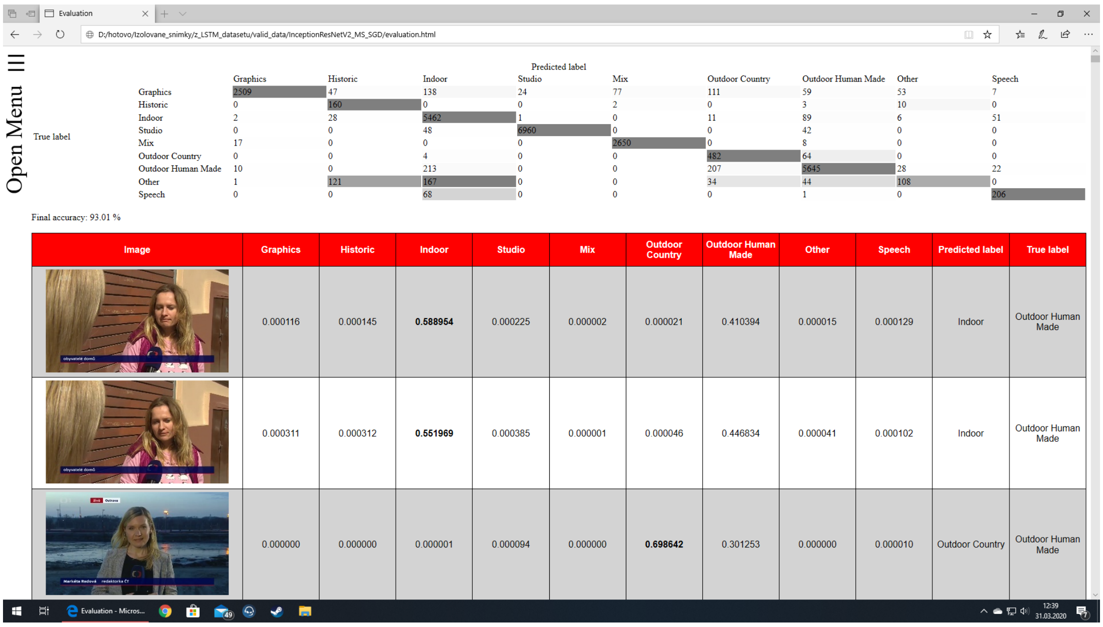
Since there is a possibility of visualizing the input data for purposes of this thesis, a tool that generates an HTML file has also been created. The tool expects a template, that contains user-defined tags. These tags are in *<? tag_name ?>* format and the tool replaces the tags with an array, a chart, an image, or a plain text. The following user-defined tag names are used and subsequently replaced as described in the list bellow:

- **classArray**: is replaced by an array with the class names in the corresponding order to the output vector of the model.
- **nav_bar**: is replaced by a navigation bar that filters the model's predictions according to the required class.
- **summary**: is replaced by a summary of predictions according to the used dataset, i.e. the timeline of predictions or the confusion matrix.
- **evaluation_classesOrModels**: is replaced by a header of the evaluation table that contains the class names or the model names.
- **evaluation_trueLabel**: is replaced by a section of the evaluation table that is used only if the vector of desired outputs is provided.

- **evaluation_predict**: is replaced by a body of the evaluation table, which contains: the predictions of the model, a class name that the model recognized the input as, and a class name that the input belongs to (only if the vector of desired outputs is provided).

In the tool, two basic functions for creating the evaluation file are implemented. The main difference is in how the dataset is passed. The first function expects a pair of inputs and desired outputs. The evaluation file generated by this function contains the confusion matrix and the correct classifications for each wrong prediction. The other function is used for the determination of model predictions and the function expects a directory of images. If the name of each image contains time in milliseconds, a timeline of predictions is also created and visualized.

The design of the template was inspired by examples from W3Schools [61]. CSS and JavaScript is used to control the modern appearance of charts and navigation bar. The evaluation table also contains the input image along with the model prediction and classification. In the case of a time-distributed model, a gif image is created. The tool can optionally create an attention map image but only if the time-distributed model is not provided. The attention map (also called the class activation map) highlights regions of the image that are important for prediction of the model. For this visualization, a Grad-CAM [52] technique from the Keras-vis library [29], that is similar to the backpropagation algorithm, is used. In the Figure 5.4, an example of the evaluation file containing the confusion matrix for the validation dataset is shown.


Evaluation
D:/hotovo/Izolovane_snimky/z_LSTM_datasetu/valid_data/InceptionResNetV2_MS_SGD/evaluation.html
Open Menu
Evaluation - Micros...
12:39
31.03.2020

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor Country	Outdoor Human Made	Other	Speech
True label	Graphics	2509	47	138	24	77	111	59	53	7
	Historic	0	160	0	0	2	0	3	10	0
	Indoor	2	28	5462	1	0	11	89	6	51
	Studio	0	0	48	6960	0	0	42	0	0
	Mix	17	0	0	0	2650	0	8	0	0
	Outdoor Country	0	0	4	0	0	482	64	0	0
	Outdoor Human Made	10	0	213	0	0	207	5645	28	22
	Other	1	121	167	0	0	34	44	108	0
	Speech	0	0	68	0	0	0	1	0	206

Final accuracy: 93.01 %

Image	Graphics	Historic	Indoor	Studio	Mix	Outdoor Country	Outdoor Human Made	Other	Speech	Predicted label	True label
obyvatelé domů	0.000116	0.000145	**0.588954**	0.000225	0.000002	0.000021	0.410394	0.000015	0.000129	Indoor	Outdoor Human Made
obyvatelé domů	0.000311	0.000312	**0.551969**	0.000385	0.000001	0.000046	0.446834	0.000041	0.000102	Indoor	Outdoor Human Made
Živě Ostrava Markéta Radová redaktorka ČT	0.000000	0.000000	0.000001	0.000094	0.000000	**0.698642**	0.301253	0.000000	0.000010	Outdoor Country	Outdoor Human Made

Figure 5.4: Example of an evaluation file.

Chapter 6

Experiments and results

At the beginning of this thesis, architectures capable of recognizing the scenes were designed. It was required to keep the output from the last convolutional layer as little as possible. The reason for this decision is to maintain a minimal number of training parameters. The structure of the created architectures was in the usage of 3 × 3 convolutional layers and 2 × 2 max-pooling layers, i.e. the same way as in the VGG networks. Then the experiments were performed by adding/removing the dropout and changing the following features:

- Parameters of convolutional layers, such as **strides, padding and number of kernels**
- **Strides**, a parameter of pooling layers.
- **Number of convolutional layers** in the blocks terminated by the pooling layer.
- **Number of dense layers and their neurons** behind all of the convolutional blocks.
- **Activation functions** of the convolutional and dense layers.
- **Optimizer** type and its parameters.

The initial models were unable to achieve accuracy of correct classification better than 0.34 for given training and validation sets. The datasets are unbalanced to maintain the distribution of individual classes as they appear in a real-world scenario. The resulting models often miss-classified all of the inputs into one class. The problem was solved by removing some of the convolutional blocks.

The first improvement was made by keeping only the first two convolutional blocks followed by fully-connected layers. In the first block a total of four convolutional layers were used (two with 16 kernels and two with 32 kernels). In the second block three convolutional layers with 64 kernels were used. Each convolutional block was terminated by the max-pooling layer with strides and dropout set to 2 and 0.2 respectively. A dense layer with 32 neurons and a dropout of 0.4 were used ahead of the classification layer. This network was able to reach an accuracy of 0.66 for given validation data. This architecture is summarized in the Table 6.1 and denoted as configuration A.

Configuration A has been further improved by changing the features mentioned above. The experiments have shown that classification accuracy can be increased by stacking convolution blocks which are terminated by the max-pooling layer. The amounts of convolutional layers were reduced in each block to make the model capable of learning. The final architecture has been created with a focus on the small amount of parameters. Each convolutional layer was immediately followed by a max-pooling layer, forming a convolutional block. A group of four of those blocks were followed by a fully-connected layer and a classification layer. This configuration is denoted as configuration B in Table 6.1 and results in accuracy of 0.72 for given validation data. After finishing this configuration, an analysis described in the following sections was conducted. This analysis was exploring the possibility of performance improvements by using an already-trained network.

Configuration	
A	B
input(180, 320, 3)	
conv(16, relu)	conv(32, relu)
conv(16, relu)	max-pool
conv(32, relu)	conv(32, relu)
conv(32, relu)	max-pool
max-pool	dropout(0.2)
dropout(0.2)	conv(64, relu)
conv(64, relu)	max-pool
conv(64, relu)	conv(64, relu)
conv(64, relu)	max-pool
max-pool	dropout(0.2)
dropout(0.2)	
flatten	
dense(32, relu)	dense(512, relu)
dropout(0.4)	dropout(0.5)
dense(9, softmax)	

Table 6.1: Configuration of created architectures, convolutional and dense layers are denoted as *<layer type>(<number of channels>, <activation function>).*

6.1 Analysis of transfer learning

The analysis of before mentioned architectures shows that the neural networks are trainable for scene type recognition. The next step led to the usage of more complex architectures. The following analysis was performed by using the VGG16 [54] with pre-trained parameters on the ImageNet dataset [49]. As the input shape of the pre-trained network is not corresponding with the shape of used dataset, the top layers were replaced by new fully-connected layers. Then new experiments were performed with changes of parameters of top layers and optimizers.

Four types of top layers were designed for the experiment. Two types contained one fully-connected layer with 700 or 1,400 neurons respectively. The other two types consist of sequence of two fully-connected layers (each with 700 or 1,400 neurons). At the end of the network, a classification layer was placed. The model with one fully-connected layer of 1,400 neurons brought the best result, with classification accuracy of 0.81 for given validation data.

Lastly, an analysis was performed before selecting the appropriate architecture for scene recognition. In this analysis, all parameters of the VGG network were randomly initialized, instead of using the pre-trained ones. The accuracy of created models was up to 0.47 for the training and validation set. These results led to the transfer learning by using pre-trained architectures from the Keras Applications [6] module.

6.2 Selecting a backbone architecture

Until now analyses, that explored the possibility of creating a scene type recognition network, were performed. It was concluded that using pre-trained networks brings more benefits in learning. In the following experiments, an appropriate optimizer was selected along with the network architectures for the classification task. The classification accuracy was used as a metric for comparison of the models and optimizers. Finally, the attention maps and classification accuracy on the test dataset are compared only for relevant models.

6.2.1 Preparing the experiments

Before comparing the different architectures, the optimizer and the top layers were selected. For purposes of these procedures, the pre-trained VGG16 network with the classification layer stacked behind all convolutional blocks, was chosen. The hyperparameters were tuned depending on the progress of learning. This preparation aims to determine a kind of baseline from which the following experiments will be based. The idea is that if there would be several relevant candidates in the selection of the backbone architecture, the parameters from this section will be fine-tuned.

First, four optimizers were nominated to train the network. During an earlier analysis, the Stochastic Gradient Descent with momentum proved very potent. From adaptive learning rate methods, RMSprop, Adam, and AMSGrad were chosen. The hyperparameters were tuned with respect to the network accuracy in each epoch. In the Figure 6.1, individual optimizers with the best-found settings are compared.

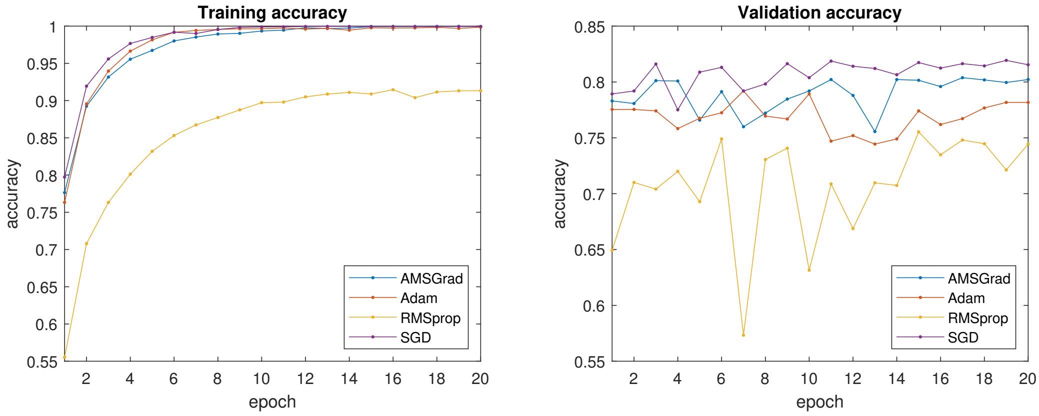


Figure 6.1: Training and validation accuracy during the learning progress for different optimizers.

According to the Figure 6.1, using RMSprop resulted in the slowest training, while the fastest training was achieved by using SGD. By comparing all optimizers on the validation dataset, RMSprop has proven to be the most unstable one. Another adjustment of optimizer parameters usually causes that learning comes to a halt and for this reason, RMSprop is inappropriate. Unlike the RMSprop, the SDG, Adam, and AMSGrad are stable. This stability is useful when selecting the optimal number of epochs for network training. The model trained by SGD had the best overall performance. For this reason, SGD was selected for the following experiments.

Initial analyses have shown that the network had better performance using one fully-connected layer than with two fully-connected layers before the classification layer. It also has been confirmed that a model with 1,400 neurons in this layer reached higher accuracy than a model with 700 neurons. Therefore a fully-connected layer with a range from 1,024 neurons to 2,816 neurons was used for analysis of top layers. The analysis of using various amount of neurons in the fully-connected layer is summarized in the Table 6.2. The evaluation took into account the final accuracy and a selective average accuracy of the validation dataset. This selective average was calculated from epochs for which the accuracy of the training set was higher than 0.999 of all used models.

Epoch	Number of neurons in fully-connected layer							
	1,024	1,280	1,536	1,792	2,048	2,304	2,560	2,816
20	0.807	0.815	0.811	**0.830**	0.816	0.818	0.822	0.821
$\bar{x}_{[13,20]}$	0.818	0.808	0.803	**0.822**	0.816	0.821	0.819	0.814

Table 6.2: Final accuracy and average accuracy (denoted as $\bar{x}$) of the last eight epochs on the validation dataset using a various number of neurons in the fully-connected layer. The subscript of the average accuracy indicates the boundary index of epochs from which the average was calculated. The full form of the table is given in the Appendix B.1.

All of the top layer settings reached higher accuracy than 0.999 on the training dataset before the 13th epoch. On the validation dataset, the first epoch accuracy was at least 0.77 for all models. This accuracy was gradually increased until the final 20th epoch. The whole learning process is shown in Appendix B.1. According to the Table 6.2, the highest accuracy was reached using the 1,792 neurons in the fully-connected layer. This configuration also reached the highest average in the last 8 epochs. Therefore, for further experiments, this top layer with one fully-connected layer in front of the classification layer was selected. Since a network with only the classification layer behind all convolution blocks proved to be effective, this top layer configuration was also used for backbone architecture selection.

6.2.2 Comparing the architectures

Although the VGG16 network has been successfully trained, the architecture does not contain skip or parallel connections, which could improve the recognition performance. Therefore, the InceptionV3 and the InceptionResNetV2 architectures were compared to the VGG16. For comparison, the MobileNetV2, which represents a low-demands network, has also been selected. The Table 6.3 summarizes the performance of all trained networks after the last epoch. The visualization of learning progress is shown in the Figure 6.2 for VGG16, InceptionV3, and InceptionResNetV2.

The Table 6.3 shows that MobileNetV2 network had the lowest number of parameters in the configuration without a fully-connected layer (denoted as F:S). By using this layer, the number of parameters grows from 3 million to 140 million. The reason is the usage of the Flatten layer, which reshapes a matrix into a vector. The output from the Flatten layer consist of 76,800 elements that are fully-connected with the 1,792 neurons. For this reason, the MobileNetV2 network is selected for the following analysis to explore the impact of using global pooling layers.

	Validation accuracy		Total parameters	
	top layer		top layer	
architecture	F:S	F:FC:S	F:S	F:FC:S
VGG16	0.815	0.830	14,945,097	60,607,817
InceptionV3	0.824	0.826	22,392,617	139,261,225
InceptionResNetV2	0.848	0.834	54,779,113	142,435,049
MobileNetV2	0.691	0.647	2,949,193	139,901,513

Table 6.3: The final accuracy of compared architectures on the validation dataset (on the left) and a total number of parameters (on the right). The top layers are denoted as: F - Flatten layer, FC - fully-connected layer, S - classification layer (dense layer with softmax function).

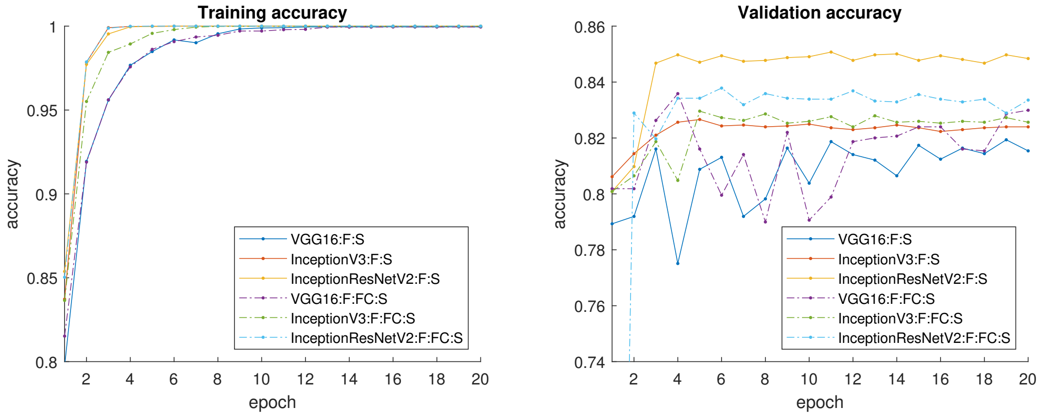


Figure 6.2: Training and validation accuracy during the learning progress for different architectures. The top layers are denoted as: F - flatten layer, FC - fully-connected layer, S - classification layer (dense layer with softmax function).

In the Figure 6.2, it can be seen that the InceptionResNetV2 has reached the highest overall accuracy for both configurations. The F:S configuration (Flatten layer followed by the classification layer) has shown better performance than the model including a fully-connected layer in front of the classification one. The final validation accuracy of InceptionResNetV2 network is 0.848 for the F:S configuration and this architecture is further analyzed along with the MobileNetV2 network.

6.2.3 Analysis of using Global Pooling layer

The Flatten layer transmits all available information to the top layers, resulting in a higher number of parameters. The target of the network design is to reduce the model complexity while maintaining or improving its performance, therefore global pooling layers can be used instead of the Flatten layer. These layers compute the average values (Global Average Pooling) or select the max values (Global Max Pooling) for each feature map of the previous layer. For the analysis of these layers, InceptionResNetV2 and MobileNetV2 architectures have been selected.

InceptionResNetV2

The InceptionResNetV2 has shown the best performance in the architectures comparison. The following Table 6.4 contains the final accuracy for given validation dataset and the total number of parameters of analyzed networks with top layers. The learning progress is shown in the Figure 6.3.

	Validation accuracy		Total parameters	
	S	FC:S	S	FC:S
Flatten	0.848	0.834	54,779,113	142,435,049
GAP	0.826	0.839	54,350,569	57,107,177
GMP	0.853	0.824	54,350,569	57,107,177

Table 6.4: The final accuracy of used top layers on validation dataset (on the left) and a total number of parameters (on the right). The comparison is performed for InceptionResNetV2 network and layers are denoted as: GAP - Global Average Pooling layer, GMP - Global Max Pooling layer, FC - fully-connected layer, S - classification layer (dense layer with softmax function).

According to the Figure 6.3, the InceptionResNetV2 including the fully-connected layer before the classification one (the FC:S configuration) has reached the best performance by using the Global Average Pooling. However, using the Global Max Pooling layer has lowered the recognition accuracy. From the Table 6.4, one can observe, that the network with the Flatten layer has 2.5 times more parameters, than a network using the global pooling layer, in this configuration. In the case of the network without the fully-connected layer, usage of a Global Max Pooling layer has shown sightly better than the usage of a Flatten layer, whereas the worst performance was gained by using a Global Average Pooling layer. The configurations without a fully-connected layer are further analyzed, because using this layer before the classification one has always proven to be defective in analyses of this network.

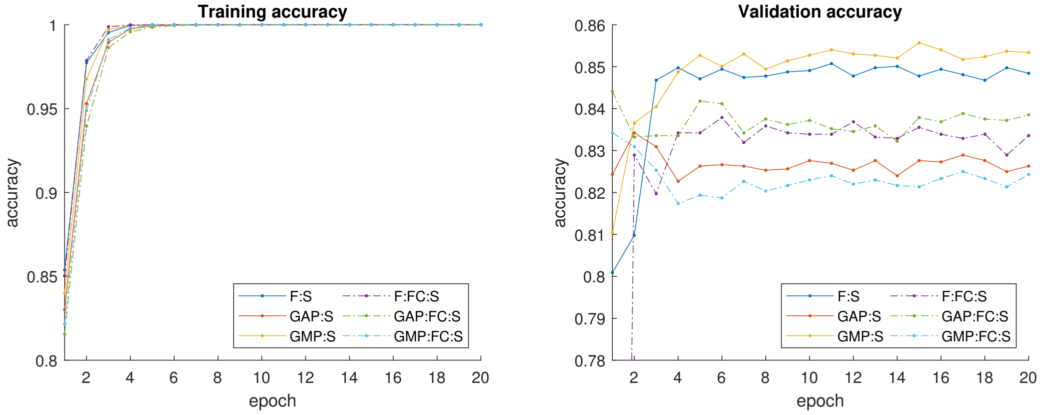


Figure 6.3: Training and validation accuracy during the learning progress for various settings of the top layers of the InceptionResNetV2. The top layers are denoted as: F - flatten layer, GAP - Global Average Pooling layer, GMP - Global Max Pooling layer, FC - fully-connected layer, S - classification layer (dense layer with softmax function).

MobileNetV2

The low demand network (MobileNetV2) did not produce satisfactory results using the Flatten layer. This leads to the analysis of using a global pooling layer. In the Figure 6.4, there are shown learning progresses and the Table 6.5 summarizes the final accuracy and the total number of parameters of the analyzed top layers.

	Validation accuracy		Total parameters	
	S	FC:S	S	FC:S
Flatten	0.691	0.645	2,949,193	139,901,513
GAP	0.843	0.838	2,269,513	4,569,673
GMP	0.693	0.315	2,269,513	4,569,673

Table 6.5: The final accuracy of used top layers on validation dataset (on the left) and a total number of parameters (on the right). The comparison is performed for MobileNetV2 network and layers are denoted as: GAP - Global Average Pooling layer, GMP - Global Max Pooling layer, FC - fully-connected layer, S - classification layer (dense layer with softmax function).

From Table 6.5 and Figure 6.4, one can see, that using a Global Max Pooling layer does not bring improvement to the recognition performance. While using the configuration with the fully-connected layer, the learning progress gets stuck and the model is unable to train further. By using the Global Average Pooling layer, the recognition accuracy has increased up to 0.843 on the validation data

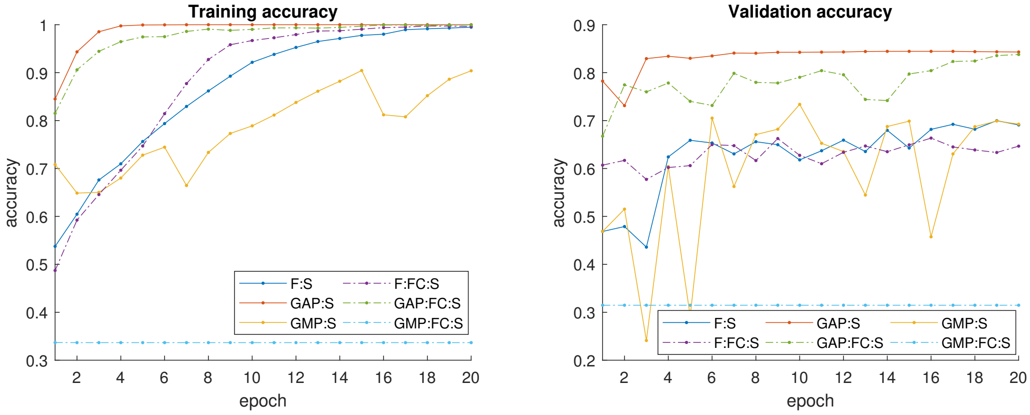


Figure 6.4: Training and validation accuracy during the learning progress for various configurations of the top layers of MobileNetV2. The top layers are denoted as: F - flatten layer, GAP - Global Average Pooling layer, GMP - Global Max Pooling layer, FC - fully-connected layer, S - classification layer (dense layer with softmax function).

and the network was able to fit on the training set. In comparison with other architectures (see Chapter 6.2.2), the MobileNetV2 (with GAP) final accuracy is higher compare to the VGG and InceptionV3 networks, and the network performance is comparable with the InceptionResNetV2. The MobileNetV2 contains far-fewer parameters than the InceptionResNetV2 (see Table 6.5 and 6.4). For these results, the MobileNetV2 with a Global Average Pooling layer is further analyzed along with the InceptionResNetV2 networks.

6.2.4 Evaluation of selected networks

In this analysis, practical usage of selected networks is simulated. The aim is to examine the influence of different architectures on the recognition performance. For purposes of this evaluation, a test dataset, which has a different density than the training and validation data, is used. The comparison has been first executed for the InceptionResNetV2 with the global pooling and Flatten layers. Then, the MobileNetV2 is compared to the most relevant InceptionResNetV2 model.

InceptionResNetV2

The classification accuracy is compared in the Table 6.6 and confusion matrices are visualized in Tables 6.7, 6.8, and 6.9 for the analyzed models. Attention maps are shown in Figures 6.5 and 6.6 for correct and incorrect classifications.

From Table 6.6, one can see that a network with a Global Average Pooling layer has weakest performance compared to the other models. The model's accuracy of validation and test data (see Table 6.4 and 6.6) is about 2 % lower than the one of

	GAP	GMP	Flatten
Accuracy	0.617	0.645	0.665

Table 6.6: Classification accuracy of the InceptionResNetV2 network using a Global Average Pooling (GAP), a Global Max Pooling (GMP), and a Flatten layer for given test data.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	O. human-made	Other	Speech
True label	Graphics	0.466	0.003	0.029	0.001	0.023	0.013	0.274	0.191	0.000
	Historic	0.347	0.600	0.000	0.000	0.000	0.000	0.053	0.000	0.000
	Indoor	0.006	0.012	0.803	0.062	0.000	0.001	0.048	0.060	0.006
	Studio	0.000	0.000	0.298	0.290	0.000	0.000	0.382	0.000	0.030
	Mix	0.012	0.000	0.367	0.094	0.304	0.000	0.224	0.000	0.000
	Outdoor country	0.000	0.000	0.000	0.001	0.000	0.553	0.411	0.033	0.001
	Outdoor human-made	0.001	0.001	0.099	0.003	0.003	0.056	0.828	0.010	0.000
	Other	0.105	0.076	0.412	0.015	0.000	0.026	0.191	0.161	0.013
	Speech	0.030	0.000	0.685	0.190	0.035	0.000	0.005	0.000	0.055

Table 6.7: Normalized confusion matrix over the test dataset of the InceptionResNetV2 network using a Global Average Pooling layer.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	O. human-made	Other	Speech
True label	Graphics	0.391	0.003	0.057	0.043	0.030	0.009	0.339	0.129	0.000
	Historic	0.000	0.920	0.000	0.000	0.000	0.000	0.080	0.000	0.000
	Indoor	0.001	0.004	0.938	0.014	0.007	0.000	0.022	0.000	0.012
	Studio	0.003	0.000	0.592	0.265	0.000	0.040	0.100	0.000	0.000
	Mix	0.000	0.000	0.433	0.033	0.202	0.000	0.332	0.000	0.000
	Outdoor country	0.000	0.000	0.019	0.000	0.000	0.537	0.434	0.000	0.010
	Outdoor human-made	0.001	0.001	0.072	0.000	0.003	0.039	0.882	0.003	0.000
	Other	0.065	0.094	0.585	0.001	0.000	0.000	0.207	0.040	0.007
	Speech	0.000	0.000	0.765	0.000	0.125	0.000	0.035	0.000	0.075

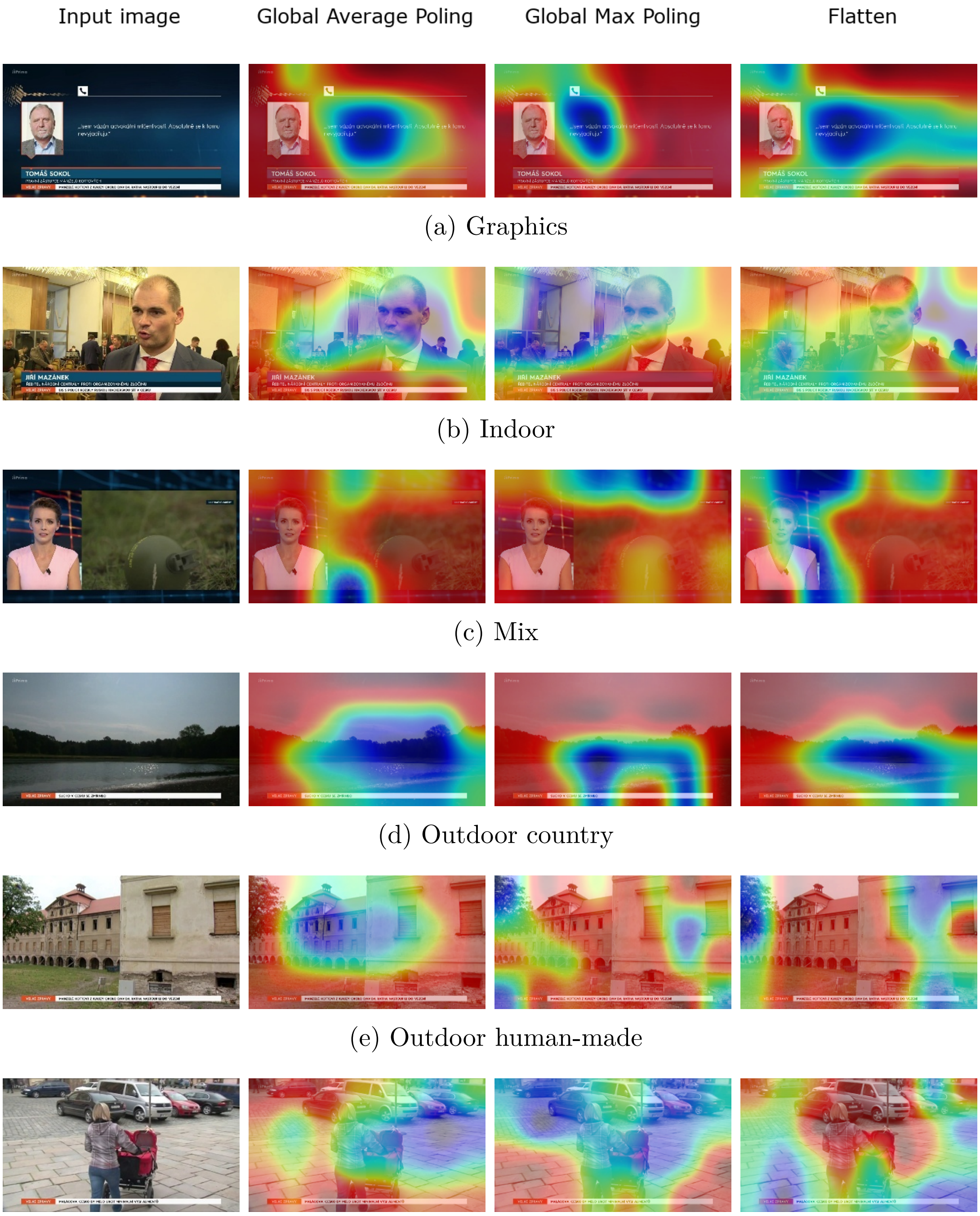
Table 6.8: Normalized confusion matrix over the test dataset of the InceptionResNetV2 network using a Global Max Pooling layer.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	O. human-made	Other	Speech
True label	Graphics	0.439	0.001	0.119	0.000	0.004	0.030	0.261	0.146	0.000
	Historic	0.280	0.720	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Indoor	0.000	0.000	0.939	0.011	0.000	0.000	0.045	0.000	0.003
	Studio	0.000	0.000	0.432	0.327	0.000	0.000	0.237	0.005	0.000
	Mix	0.000	0.000	0.504	0.054	0.249	0.000	0.193	0.000	0.000
	Outdoor country	0.001	0.000	0.001	0.000	0.000	0.662	0.335	0.000	0.000
	Outdoor human-made	0.001	0.003	0.084	0.002	0.000	0.050	0.859	0.001	0.000
	Other	0.051	0.026	0.579	0.003	0.000	0.030	0.188	0.124	0.000
	Speech	0.060	0.000	0.905	0.000	0.000	0.000	0.000	0.000	0.035

Table 6.9: Normalized confusion matrix over the test dataset of the Inception-ResNetV2 network using a Flatten layer.

a network using a Flatten or a Global Max Pooling layer. As mentioned earlier, datasets are not balanced in order to maintain the distribution of recognized scenes in the broadcasts. In TV News, the Indoor and both of the Outdoor scenes appear most often, for which the network (with a Global Average Pooling layer) more often misclassified than the other models. Contrarily, the model with a Global Max Pooling layer was less successful in classifying Studio and Mix scenes.

By comparing the attention maps of correct classifications (Figure 6.5), we can observe in the case of Mix class that the models with a Global Average Pooling and a Flatten layer are more focused on recognizing two different scenes than the model with a Global Max Pooling layer. In an Outdoor human-made scenes, the model with a Flatten layer is better focused on more informative elements, than models with a global pooling layer, on both examples. The model with a Global Max Polling layer is watching boundary parts of the image in the example with a building, whereas other models are focused on the whole building in the middle of the image. In the second example, the model with a Global Average Pooling is watching only for the sidewalk, while the other models are also focusing on the cars. In Figure 6.6, one can observe that although the models classified wrongly, their attentions are focused on sensible sections of the input image. For example, the model with a Global Max Pooling layer is watching for a microphone when it is predicting to the Speech scene (see Indoor scene). For the Speech scene example, the model with a Flatten layer is not looking at the character when it is classifying the input image to the Graphics class.



(a) Graphics

(b) Indoor

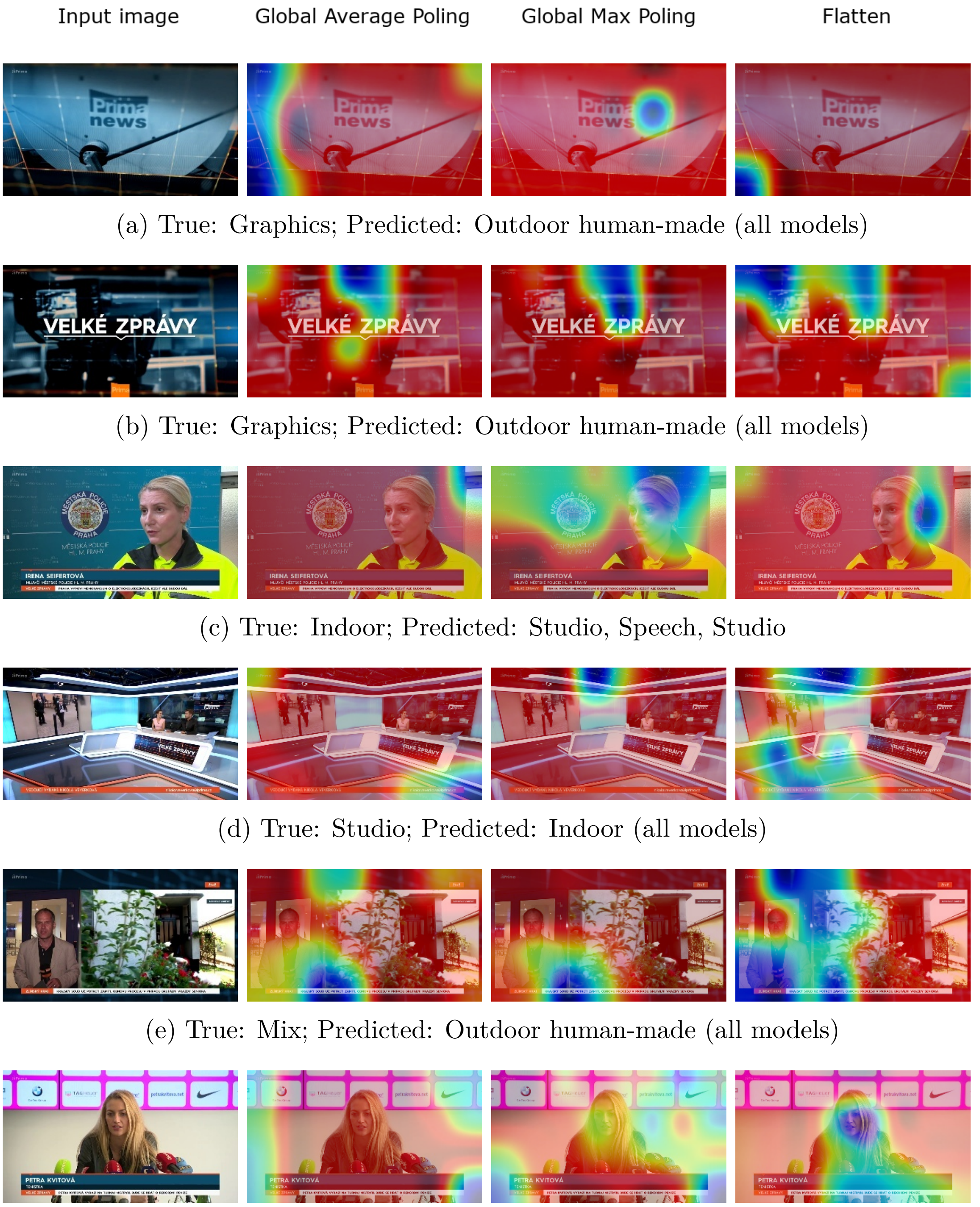
(c) Mix

(d) Outdoor country

(e) Outdoor human-made

(f) Outdoor human-made

Figure 6.5: Visualization of correct classifications of examples from the test dataset. Input images are in the leftmost column, followed by GradCAM visualizations of used models. The comparison is performed for the InceptionResNetV2 network with a Global Average Pooling layer (center-left column), a Global Max Pooling layer (center-right column), and a Flatten layer (rightmost column).



(a) True: Graphics; Predicted: Outdoor human-made (all models)

(b) True: Graphics; Predicted: Outdoor human-made (all models)

(c) True: Indoor; Predicted: Studio, Speech, Studio

(d) True: Studio; Predicted: Indoor (all models)

(e) True: Mix; Predicted: Outdoor human-made (all models)

(f) True: Speech; Predicted: Indoor, Mix, Graphics

Figure 6.6: Visualization of incorrect classifications of examples from the test dataset. Input images are in the leftmost column, followed by GradCAM visualizations of used models. The comparison is performed for the InceptionResNetV2 network with a Global Average Pooling layer (center-left column), a Global Max Pooling layer (center-right column), and a Flatten layer (rightmost column).

MobileNetV2

The classification accuracy of MobileNetV2 models using various top layers are shown in Table 6.10 for given test dataset. The confusion matrices are visualized in Tables 6.11 and 6.12. In the Figures 6.7 and 6.8 attention maps of correct and incorrect classifications are compared.

	Top layers	
	GAP:S	GAP:FC:S
Accuracy	0.667	0.675

Table 6.10: Classification accuracy for given test data of the MobileNetV2 network using different top layers. The top layers are denoted as: GAP - Global Average Pooling layer, FC - fully-connected layer, S - classification layer (dense layer with softmax function).

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	O. human-made	Other	Speech
True label	Graphics	0.576	0.074	0.094	0.000	0.067	0.023	0.066	0.100	0.000
	Historic	0.373	0.627	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Indoor	0.026	0.026	0.893	0.003	0.001	0.001	0.040	0.000	0.009
	Studio	0.000	0.007	0.553	0.225	0.000	0.000	0.215	0.000	0.000
	Mix	0.012	0.000	0.494	0.000	0.259	0.000	0.235	0.000	0.000
	Outdoor country	0.007	0.000	0.000	0.000	0.000	0.774	0.219	0.000	0.000
	Outdoor human-made	0.014	0.012	0.074	0.007	0.000	0.049	0.842	0.001	0.000
	Other	0.214	0.021	0.300	0.006	0.000	0.033	0.242	0.150	0.034
	Speech	0.000	0.000	0.805	0.000	0.000	0.000	0.000	0.000	0.195

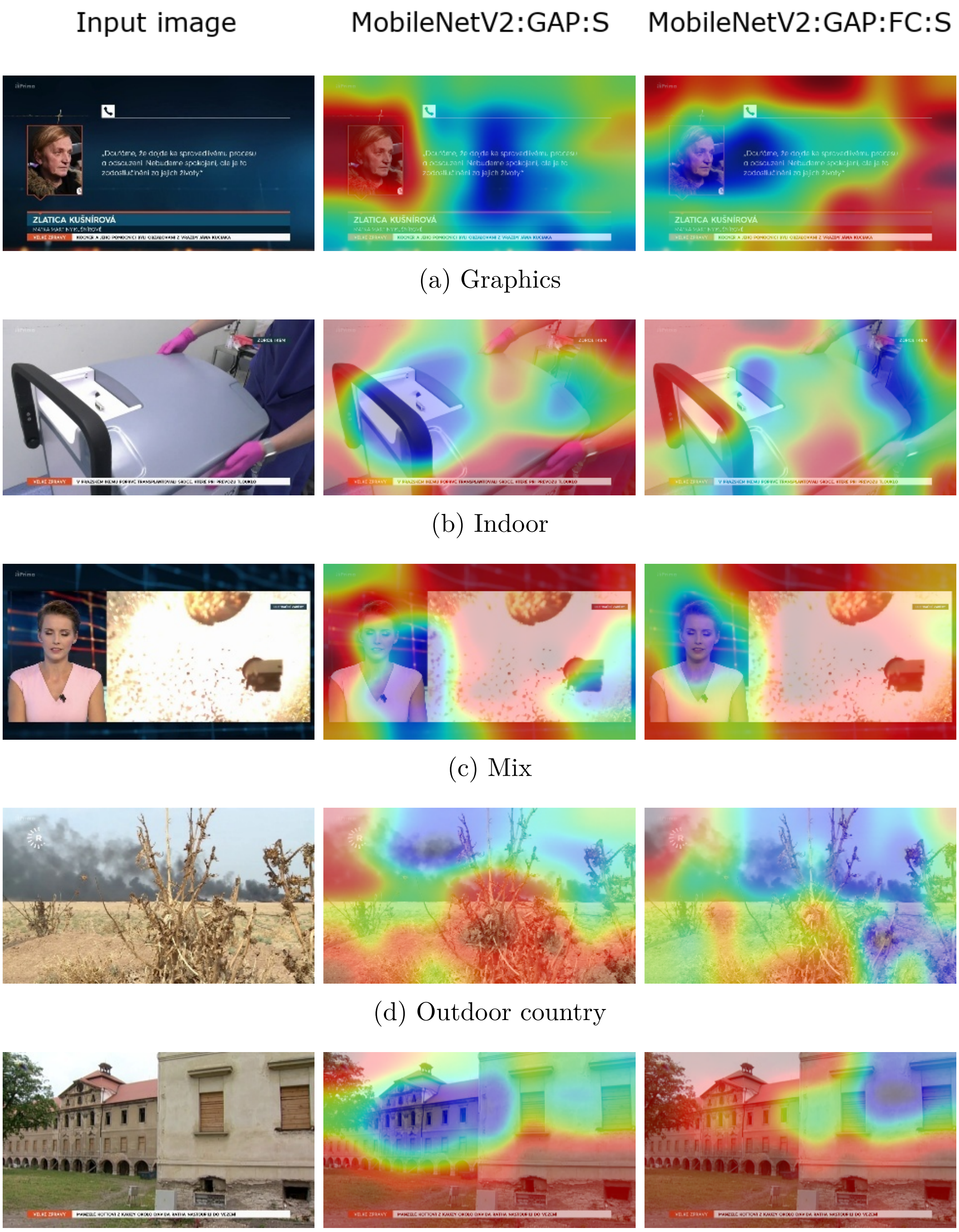
Table 6.11: Normalized confusion matrix over the test dataset of the MobileNetV2 network using a Global Average Pooling layer followed by a classification layer.

By comparing Table 6.10 and 6.6, one can observe, the classification accuracies of MobileNetV2 models are a little bit higher than in the case of the analyzed InceptionResNetV2 networks. From confusion matrices (see Table 6.11 and 6.12), one can see, that the GAP:FC:S configuration (a Global Average Pooling layer followed by a fully connected and classification layer) of the MobileNetV2 network classifies to the Mix, Other, and Speech classes better than the model with the GAP:S configuration. However, the model with GAP:FC:S configuration was rarely classifying to the Studio class in comparison with the second model (i.e. GAP:S configuration).

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	O. human-made	Other	Speech
True label	Graphics	0.531	0.059	0.244	0.000	0.024	0.029	0.109	0.004	0.000
	Historic	0.013	0.987	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Indoor	0.025	0.013	0.892	0.000	0.000	0.003	0.027	0.003	0.036
	Studio	0.000	0.042	0.863	0.032	0.000	0.000	0.063	0.000	0.000
	Mix	0.000	0.000	0.280	0.000	0.628	0.000	0.092	0.000	0.000
	Outdoor country	0.001	0.000	0.000	0.000	0.000	0.714	0.269	0.015	0.000
	Outdoor human-made	0.004	0.008	0.076	0.000	0.000	0.069	0.830	0.012	0.001
	Other	0.211	0.074	0.318	0.000	0.000	0.001	0.120	0.258	0.019
	Speech	0.000	0.000	0.705	0.000	0.000	0.000	0.000	0.000	0.295

Table 6.12: Normalized confusion matrix over the test dataset of the MobileNetV2 network using a Global Average Pooling layer followed by fully-connected and classification layers.

From attention maps of correct classifications (Figure 6.7), one can observe that in case of the Graphics class, the model with the GAP:S configuration of top layers is focused on a photo on the left in the image, while the second model (GAP:FC:S) is more looking for the boundary parts of the input. In the Indoor example, both models are correctly more focused on the background parts that provides the main information about being inside the building. In the example of the Outdoor human-made scene, the model with GAP:FC:S configuration is observing the whole left-side building, while the model with GAP:S configuration is looking for the bottom of the input image. In the case of the incorrect classifications (Figure 6.8), both models incorrectly classified the example of the Mix scene, in which the moderator is on the other side of the image than the models saw in the training data. Both models also misclassified the Outdoor country example which does not contain any building or vehicle, or the Other scene example that consists of lights and cables coming from a computer.



(a) Graphics

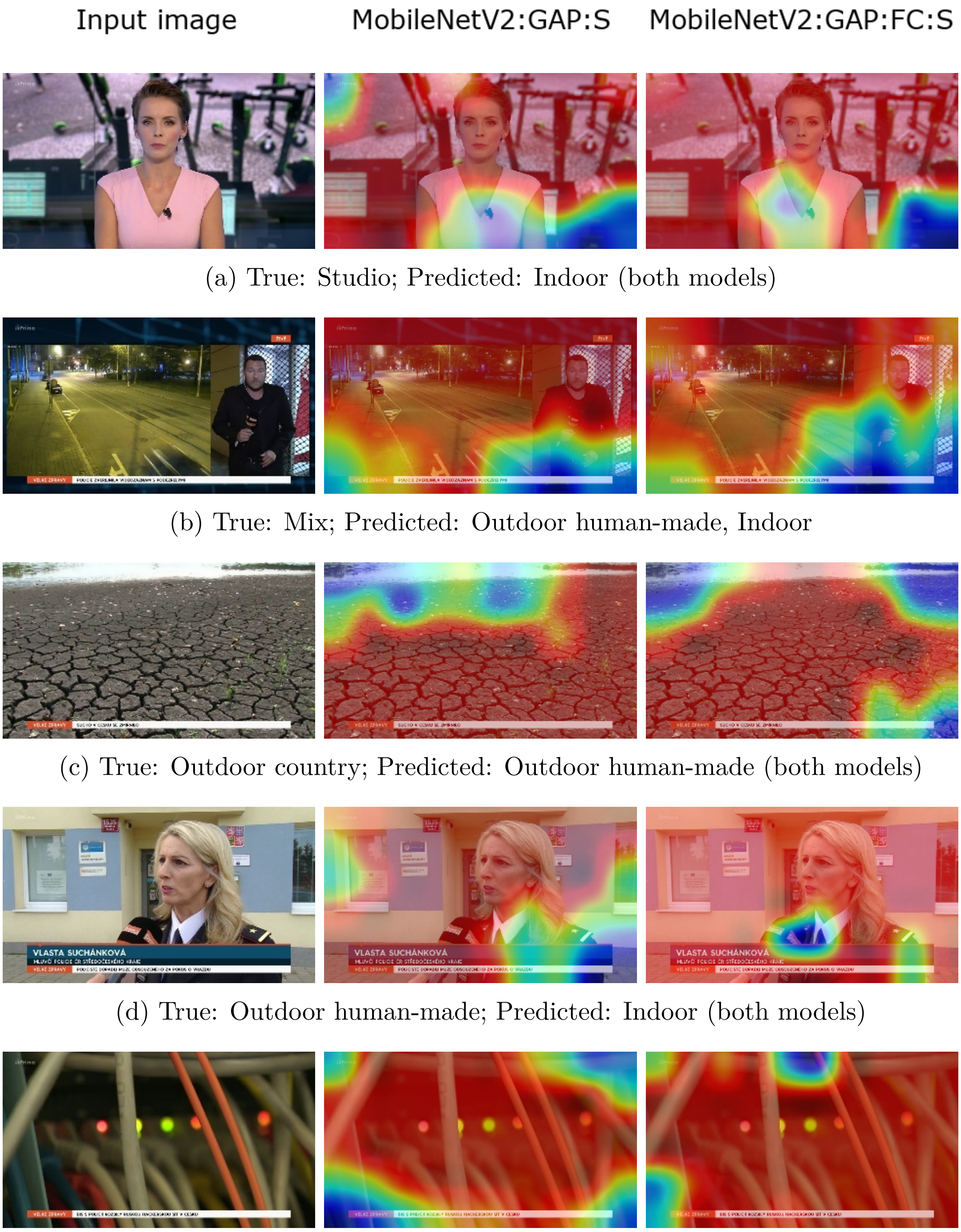
(b) Indoor

(c) Mix

(d) Outdoor country

(e) Outdoor human-made

Figure 6.7: Visualization of correct classifications of examples from the test dataset. Input images are in the left column, followed by a GradCAM visualizations of used models. The comparison is performed for the MobileNetV2 network with different top layers configuration. The top layers are denoted as: GAP - Global Average Pooling layer, FC - fully-connected layer, S - classification layer (dense layer with a softmax function).



(a) True: Studio; Predicted: Indoor (both models)

(b) True: Mix; Predicted: Outdoor human-made, Indoor

(c) True: Outdoor country; Predicted: Outdoor human-made (both models)

(d) True: Outdoor human-made; Predicted: Indoor (both models)

(e) True: Other; Predicted: Outdoor human-made, Indoor

Figure 6.8: Visualization of incorrect classifications of examples from the test dataset. Input images are in the left column, followed by a GradCAM visualizations of used models. The comparison is performed for the MobileNetV2 network with different top layers configuration. The top layers are denoted as: GAP - Global Average Pooling layer, FC - fully-connected layer, S - classification layer (dense layer with a softmax function).

6.3 Time Distributed models

In this section, the possibility of using selected models for processing time-series data is analyzed. The backbone architecture of the models is a Inception-ResNetV2 or MobileNetV2 network. The classification layer was removed from the trained models and replaced by an LSTM layer along with a new classification layer. The optimizer is selected individually for InceptionResNetV2 and MobileNetV2 models. Then an analysis, which explores a possibility of using different numbers of units in the LSTM layer, is performed. Finally, the remaining models are trained and all models are evaluated for given data.

6.3.1 Optimizer selection and adjustment of an LSTM layer

First, the optimizers for networks, which are processing time distributed data, are selected. As mentioned earlier, this selection is performed individually for InceptionResNetV2 and MobileNetV2 models. The last layer from the model is replaced by an LSTM layer with 32 units, which is followed by the classification layer. In Figures 6.9 and 6.10, optimizers for the InceptionResNetV2 and MobileNetV2 networks are compared. Then the analysis, which explores the usage of the various number of LSTM units, is made. The learning progress of this analysis is shown in Figure 6.11 for the MobileNetV2 network.

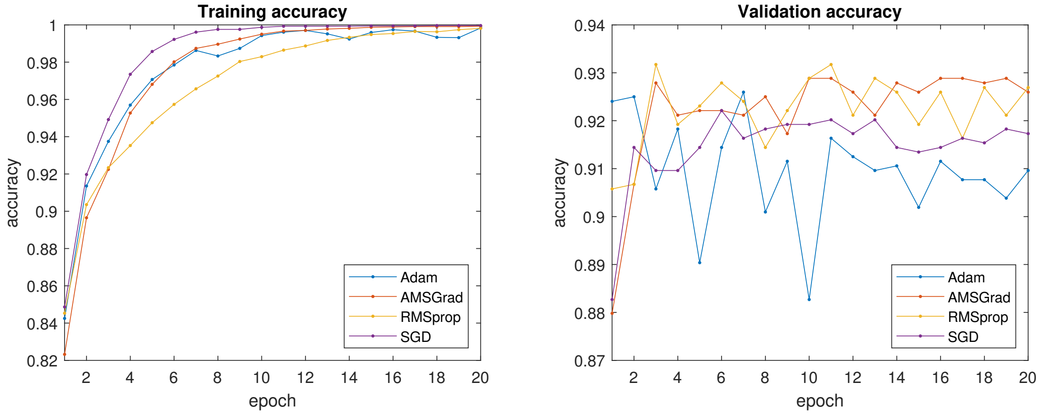


Figure 6.9: Training and validation accuracy during the learning progress of the InceptionResNetV2 network for different optimizers.

According to Figure 6.9, the performance of the InceptionResNetV2 network using Adam has shown to be inadequate in comparison with other optimizers. Overall, the best results were obtained using AMSGrad and RMSprop optimizers. AMSGrad is selected for the InceptionResNetV2 models due to the smoother learning process for given validation data. In Figure 6.10, one can observe that

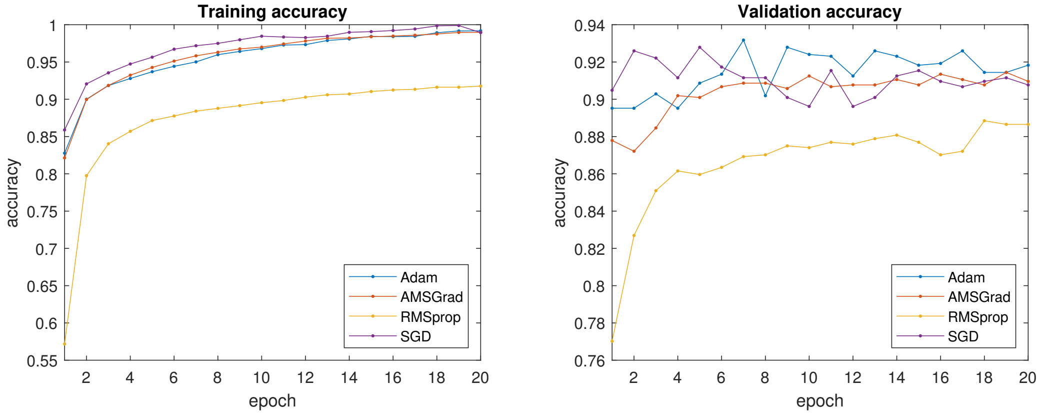


Figure 6.10: Training and validation accuracy during the learning progress of the MobileNetV2 network for different optimizers.

the learning of MobileNetV2 is the slowest for given validation data by using RMSprop optimizer. A network using SGD, whose validation accuracy is higher in early stages of learning, tends to overfit a bit faster in comparison with other optimizers. The MobileNetV2 network using Adam reached the overall highest accuracy and for this reason, Adam optimizer is selected for the MobileNetV2 models.

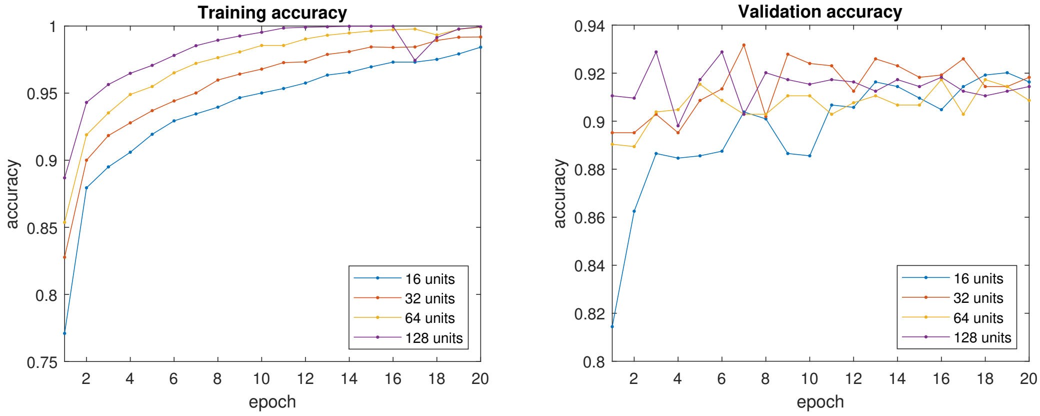


Figure 6.11: Training and validation accuracy during the learning progress for various amount of LSTM units.

The various numbers of LSTM units (see Figure 6.11) have proved that with increasing numbers of units, the model is fitting faster for given training data. In the case of validation accuracy, a model with 16 units is learning slowly until the 13th epoch, when its performance becomes similar to the other models. The model with 64 units, since the 3rd epoch, is maintaining accuracy in the range between 0.90 and 0.92. A network using 32 units is able to reach higher accuracy than the other configurations. Therefore, a configuration with 32 units in an LSTM layer have been selected.

6.3.2 Comparing the relevant models

A total of five best performing models, in terms of recognition of isolated images, were modified to process time-distributed data. The last layer from the model was replaced by an LSTM and a new classification layer. According to the analysis from the previous section, 32 units of the LSTM layer are used in all models. In Figure 6.12, learning progress of used networks is compared. Validation accuracy at the end of the learning of each model is summarized in Table 6.13.

	InceptionResNetV2		
Top layers	GAP:LSTM:S	GMP:LSTM:S	F:LSTM:S
Accuracy	0.891	0.924	0.926
Total parameters	54,537,865	54,537,865	60,632,713

	MobileNetV2	
Top layers	GAP:LSTM:S	GAP:FC:LSTM:S
Accuracy	0.918	0.947
Total parameters	2,426,345	4,787,433

Table 6.13: The final accuracy of used models on the validation dataset (on the left) and a total number of parameters (on the right). The top layers are denoted as: GAP - Global Average Pooling layer, GMP - Global Max Pooling layer, F - Flatten layer, FC - fully-connected layer, LSTM - Long short-term memory layer, S - classification layer (dense layer with softmax function).

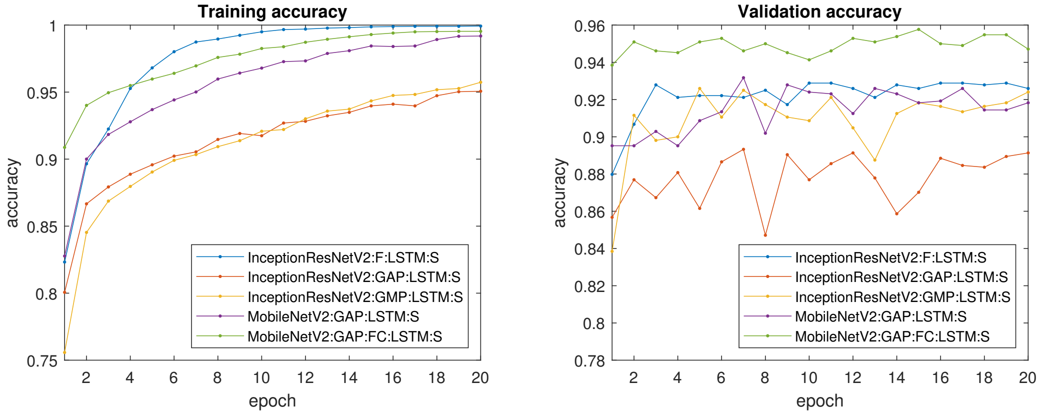


Figure 6.12: Training and validation accuracy during the learning progress for different time-distributed networks.

According to Figure 6.12, the InceptionResNetV2 network with a Global Average Pooling layer has the worst performance during the learning process for given validation data. The other configurations of the inception network have a similar performance as the MobileNetV2 network with a Global Average Pooling followed by an LSTM and the classification layer, which contains a total of 2,426,345 parameters. The MobileNetV2 network using a fully-connected layer in front of the LSTM layer reached the highest overall accuracy in comparison with other models. The final accuracy of this model is 0.947 for given validation data.

6.3.3 Evaluation of the trained models

Similarly to the selection of the backbone architecture, the time-distributed models are evaluated for given test data. The total number of images is the same for both datasets and the sets differ only in the size of the sequence of the input images. The evaluated time-distributed networks expect a series of 25 images on the input. In Table 6.19, models have been compared for given test data. Confusion matrix is available in Tables 6.14, 6.15, 6.16 for the InceptionResNetV2 networks, and in Tables 6.17 and 6.18 for the MobileNetV2 networks.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	O. human-made	Other	Speech
True label	Graphics	0.536	0.000	0.000	0.071	0.179	0.000	0.143	0.071	0.000
	Historic	0.333	0.667	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Indoor	0.024	0.012	0.659	0.098	0.037	0.000	0.049	0.012	0.110
	Studio	0.000	0.000	0.083	0.417	0.083	0.042	0.292	0.000	0.083
	Mix	0.000	0.000	0.000	0.118	0.882	0.000	0.000	0.000	0.000
	Outdoor country	0.069	0.000	0.000	0.069	0.034	0.655	0.103	0.000	0.069
	Outdoor human-made	0.037	0.009	0.028	0.092	0.064	0.055	0.697	0.018	0.000
	Other	0.219	0.062	0.281	0.156	0.031	0.031	0.062	0.125	0.031
	Speech	0.125	0.000	0.125	0.250	0.125	0.000	0.000	0.000	0.375

Table 6.14: Normalized confusion matrix over the test dataset of the InceptionResNetV2 network using a Global Average Pooling layer.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	O. human-made	Other	Speech
True label	Graphics	0.536	0.000	0.000	0.250	0.036	0.036	0.107	0.036	0.000
	Historic	0.333	0.333	0.000	0.000	0.000	0.000	0.333	0.000	0.000
	Indoor	0.012	0.000	0.890	0.012	0.037	0.000	0.024	0.000	0.024
	Studio	0.000	0.000	0.375	0.458	0.000	0.042	0.125	0.000	0.000
	Mix	0.000	0.000	0.118	0.059	0.765	0.000	0.059	0.000	0.000
	Outdoor country	0.000	0.000	0.034	0.000	0.000	0.517	0.448	0.000	0.000
	Outdoor human-made	0.018	0.009	0.037	0.000	0.046	0.046	0.844	0.000	0.000
	Other	0.250	0.062	0.312	0.031	0.000	0.000	0.219	0.031	0.094
	Speech	0.000	0.000	0.375	0.000	0.250	0.000	0.000	0.000	0.375

Table 6.15: Normalized confusion matrix over the test dataset of the Inception-ResNetV2 network using a Global Max Pooling layer.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	O. human-made	Other	Speech
True label	Graphics	0.321	0.000	0.143	0.000	0.036	0.036	0.321	0.143	0.000
	Historic	0.000	0.667	0.000	0.000	0.000	0.000	0.000	0.333	0.000
	Indoor	0.000	0.012	0.902	0.024	0.012	0.012	0.012	0.024	0.000
	Studio	0.000	0.000	0.208	0.625	0.000	0.042	0.125	0.000	0.000
	Mix	0.000	0.000	0.471	0.059	0.471	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.000	0.000	0.000	0.000	0.586	0.379	0.034	0.000
	Outdoor human-made	0.018	0.000	0.046	0.000	0.009	0.046	0.862	0.018	0.000
	Other	0.125	0.000	0.406	0.000	0.000	0.031	0.156	0.281	0.000
	Speech	0.000	0.000	0.875	0.000	0.000	0.000	0.125	0.000	0.000

Table 6.16: Normalized confusion matrix over the test dataset of the Inception-ResNetV2 network using a Flatten layer.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	O. human-made	Other	Speech
True label	Graphics	0.750	0.071	0.000	0.107	0.071	0.000	0.000	0.000	0.000
	Historic	0.333	0.667	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Indoor	0.061	0.012	0.646	0.037	0.085	0.012	0.049	0.000	0.098
	Studio	0.000	0.042	0.208	0.542	0.000	0.000	0.208	0.000	0.000
	Mix	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
	Outdoor country	0.069	0.000	0.000	0.034	0.000	0.724	0.172	0.000	0.000
	Outdoor human-made	0.055	0.018	0.009	0.064	0.073	0.028	0.743	0.000	0.009
	Other	0.375	0.062	0.156	0.125	0.000	0.000	0.156	0.062	0.062
	Speech	0.000	0.000	0.125	0.750	0.125	0.000	0.000	0.000	0.000

Table 6.17: Normalized confusion matrix over the test dataset of the MobileNetV2 network using a Global Average Pooling layer followed by LSTM and classification layers.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	O. human-made	Other	Speech
True label	Graphics	0.714	0.071	0.071	0.000	0.000	0.000	0.036	0.107	0.000
	Historic	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Indoor	0.012	0.037	0.817	0.012	0.000	0.000	0.037	0.012	0.073
	Studio	0.000	0.083	0.750	0.042	0.083	0.000	0.042	0.000	0.000
	Mix	0.000	0.000	0.294	0.000	0.588	0.000	0.118	0.000	0.000
	Outdoor country	0.034	0.000	0.000	0.000	0.000	0.448	0.483	0.034	0.000
	Outdoor human-made	0.000	0.009	0.046	0.000	0.000	0.028	0.890	0.028	0.000
	Other	0.219	0.094	0.188	0.000	0.000	0.000	0.125	0.375	0.000
	Speech	0.000	0.000	0.750	0.000	0.000	0.000	0.000	0.000	0.250

Table 6.18: Normalized confusion matrix over the test dataset of the MobileNetV2 network using a Global Average Pooling layer followed by fully-connected, LSTM, and classification layers.

	InceptionResNetV2		
Top layers	GAP:LSTM:S	GMP:LSTM:S	F:LSTM:S
Accuracy	0.596	0.675	0.687

	MobileNetV2	
Top layers	GAP:LSTM:S	GAP:FC:LSTM:S
Accuracy	0.635	0.667

Table 6.19: Classification accuracy for given test data of the InceptionResNetV2 and MobileNetV2 networks using different top layers. The top layers are denoted as: GAP - Global Average Pooling layer, GMP - Global Max Pooling layer, F - Flatten layer, FC - fully-connected layer, LSTM - Long short-term memory layer, S - classification layer (dense layer with softmax function).

According to Table 6.19, the InceptionResNetV2 networks using either a Flatten or a Global Max Pooling layer reached the highest accuracy in comparison with other models. The MobileNetV2 and InceptionResNetV2 networks using Global Average Pooling layers decreased their performance by processing sequence data (see Tables 6.6, 6.10), while an LSTM layer has shown improvements for the InceptionResNetV2 networks using both a Global Max Pooling or Flatten layer. The InceptionResNetV2 network using a Global Average Pooling layer strives to classify the input into all classes (see Table 6.14) but its classification accuracy is the lowest in comparison with other models. Both of the MobileNetV2 networks, which are processing time-series data, decreased theirs performance for frequent scenes (i.e. Indoor and both of the Outdoor classes) in comparison with their equivalent models, which are processing isolated images. The MobileNetV2 network without the fully-connected layer in front of the LSTM layer and the InceptionResNetV2 networks using a Global Max Pooling and Flatten layer sightly improved their successful classification rate for Graphics, Studio, and Mix classes (see Tables 6.17, 6.15, 6.16, and Tables 6.11, 6.8, 6.9).

6.4 Final evaluation and discussion

The application of the LSTM layer has shown improvements in the recognition of less frequent classes for given test data. However, the time-distributed models deteriorated the performance in recognition of more frequent classes and overall accuracy decreased for all models using the Global Average Pooling layer. Therefore, all of the selected single-image processing models are compared with their counterpart sequence-processing models, on the validation dataset, which was used in the training of the time-distributed networks. This comparison is shown in the following Table 6.20.

	InceptionResNetV2		
Top layers	GAP:S	GMP:S	F:S
Accuracy	0.926	0.930	0.934
Total parameters	54,350,569	54,350,569	54,779,113
Top layers	GAP:LSTM:S	GMP:LSTM:S	F:LSTM:S
Accuracy	0.891	0.924	0.926
Total parameters	54,537,865	54,537,865	60,632,713

	MobileNetV2	
Top layers	GAP:S	GAP:FC:S
Accuracy	0.943	0.942
Total parameters	2,269,513	4,569,673
Top layers	GAP:LSTM:S	GAP:FC:LSTM:S
Accuracy	0.918	0.947
Total parameters	2,426,345	4,787,433

Table 6.20: The accuracy of used models for given validation dataset and a total number of parameters. The top layers are denoted as: GAP - Global Average Pooling layer, GMP - Global Max Pooling layer, F - Flatten layer, FC - fully-connected layer, LSTM - Long short-term memory layer, S - classification layer (dense layer with softmax function).

From the obtained results (see Table 6.20), one can conclude, that a network, which is processing input data per image, brings higher overall recognition accuracy than the models, which are using an LSTM layer to process sequences of images. An LSTM layer improved the performance only in the case of the MobileNetV2 network including a fully-connected layer. One can also deduce, that the MobileNetV2 networks proved to be better (except in one case) than the complex InceptionResNetV2 networks. Due to the computational demands, the MobileNetV2 network is also more appropriate than the InceptionResNetV2 network for practical use, e.g. online processing. Therefore, the MobileNetV2 with a Global Average Pooling layer followed by the classification layer is the best configuration for processing of the isolated images. Eventually, the MobileNetV2 with a Global Average Pooling layer followed by a fully-connected layer, an LSTM layer and classification layer is the best configuration for processing sequences of images.

Chapter 7

Conclusion

This thesis deals with the analysis and creation of a neural network, which is capable of recognizing scenes of television news of Czech TV. The scene types are divided into nine categories, e.g. Graphics, Indoor, Studio. A tool, which generates an HTML file for visualizing attention maps and model's predictions for each input image, has also been created for evaluation of the network performance. The system utilizing Cut Detector was designed for practical application. The Cut Detector provides data of one continuous scene to a recognizer, which classifies a sequence of images into one class.

The obtained results reveal an impressive performance of a complex InceptionResNetV2 architecture, compared to the other models, during the learning phase for given training and validation data. In the testing phase, models using this architecture misclassified more often than models using a MobileNetV2 network as a backbone architecture, which is less computationally demanding. This suggests that models using the InceptionResNetV2 architecture have been overfitted on given training data. Analyses also show that networks, which are not processing sequences of images, have better recognition accuracy in most cases than time-distributed networks using an LSTM layer. Therefore, a network, which is not processing time-series data, should be applied along with a filter suppressing misclassifications of the network. For example, the sequence of images provided by the Cut Detector could be classified into the class, for which the neural network classified most of the individual input images. Since the accuracy classifying into 9 classes was around 94 %, which is a very promising result that allows the possibility for practical usage.

In some cases, the correct classification of the input image cannot be determined into defined classes. This happens when it is not clear whether, for example, a scene is recorded inside or outside the building. Even though the sports news is separated from television news on the channel of Czech TV, sports broadcasts sometimes appear in this show. Therefore, the Sport class could be included for purposes of TV News scenes recognition along with other classes motivated by real-world applications, e.g. government session recordings. In the future works, the recognized scenes could be further analyzed and processed, e.g. the sports broadcasts can be divided into various sport categories, or the speech could be automatically subtitled for Speech and Studio scenes.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Backpropagation. *Brilliant.org*. Retrieved 12:00, December 18, 2019. `https://brilliant.org/wiki/backpropagation/`.

[3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[4] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[5] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

[6] François Chollet et al. Keras. `https://keras.io`, 2015.

[7] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[8] J.S. Cramer. The origins of logistic regression. Tinbergen Institute Discussion Papers 02-119/4, Tinbergen Institute, December 2002.

[9] Timothy Dozat. Incorporating nesterov momentum into adam. *ICLR Workshop*, 2016.

[10] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[11] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[12] Felix A Gers, Douglas Eck, and Jürgen Schmidhuber. Applying lstm to time series predictable through time-window approaches. In *Neural Nets WIRN Vietri-01*, pages 193–200. Springer, 2002.

[13] Felix A Gers and E Schmidhuber. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001.

[14] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[15] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[17] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[18] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 273–278. IEEE, 2013.

[19] Alex Graves, Marcus Liwicki, Horst Bunke, Jürgen Schmidhuber, and Santiago Fernández. Unconstrained on-line handwriting recognition with recurrent neural networks. In *Advances in neural information processing systems*, pages 577–584, 2008.

[20] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405:947–951, 2000.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[23] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[24] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[25] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[26] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[27] Robin Jia and Percy Liang. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622*, 2016.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Raghavendra Kotikalapudi and contributors. keras-vis. `https://github.com/raghakot/keras-vis`, 2017.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[31] Haoning Lin, Zhenwei Shi, and Zhengxia Zou. Maritime semantic labeling of optical remote sensing images with multi-scale fully convolutional network. *Remote sensing*, 9(5):480, 2017.

[32] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[33] Tao Liu, Wengang Zhou, and Houqiang Li. Sign language recognition with long short-term memory. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 2871–2875. IEEE, 2016.

[34] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[35] Jindřich Matoušek and Jakub Vít. Improving automatic dubbing with subtitle timing optimisation using video cut detection. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2385–2388. IEEE, 2012.

[36] Masakazu Matsugu, Katsuhiko Mori, Yusuke Mitari, and Yuji Kaneda. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 16(5-6):555–559, 2003.

[37] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[38] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[39] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[40] Takeshi Mita, Toshimitsu Kaneko, and Osamu Hori. Joint haar-like features for face detection. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1619–1626. IEEE, 2005.

[41] L. Müller. *Automatické titulkování živých pořadů České televize – současný stav a výhled do budoucna.* Report: INSPO 2012. Prague, Czech Republic, 2012.

[42] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[43] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*, 7, 2017.

[44] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[45] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para.* Cornell Aeronautical Laboratory, 1957.

[46] Frank Rosenblatt. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms.* Spartan Books, New York, 1961.

[47] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323:533–536, 1986.

[48] Olga Russakovsky, Jia Deng, Zhiheng Huang, Alexander C. Berg, and Li Fei-Fei. Detecting avocados to zucchinis: what have we done, and where are we going? In *International Conference on Computer Vision (ICCV)*, 2013.

[49] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[50] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014.

[51] Frank Seide and Amit Agarwal. Cntk: Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2135–2135, 2016.

[52] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[53] Hoo-Chang Shin, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285–1298, 2016.

[54] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[55] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[56] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[57] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[58] The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[59] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[60] Martijn van Otterlo and Marco Wiering. *Reinforcement learning and markov decision processes*. Springer, 2012.

[61] W3Schools. *W3Schools Online Web Tutorials*. 2020. `http://www.w3schools.com`.

[62] Le Wang, Xuhuan Duan, Qilin Zhang, Zhenxing Niu, Gang Hua, and Nanning Zheng. Segment-tube: Spatio-temporal action localization in untrimmed videos with per-frame segmentation. *Sensors*, 18(5):1657, 2018.

[63] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[64] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. 2020. `https://d2l.ai`.

Appendices

Appendix A

Image examples

A.1 Examples from the Czech Television News

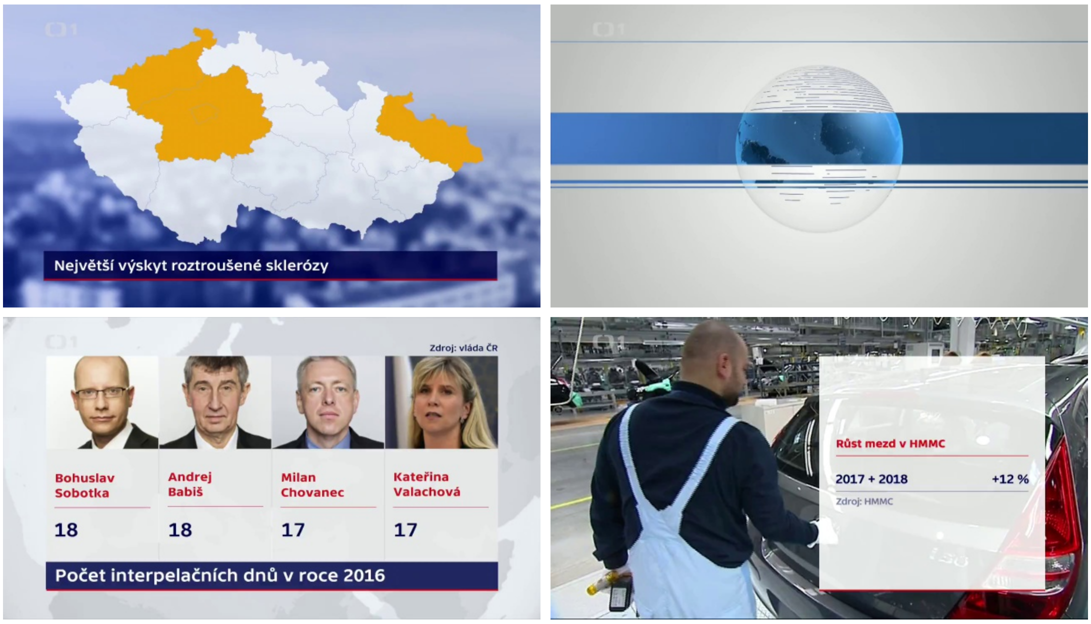


Figure A.1: Examples for a Graphics class

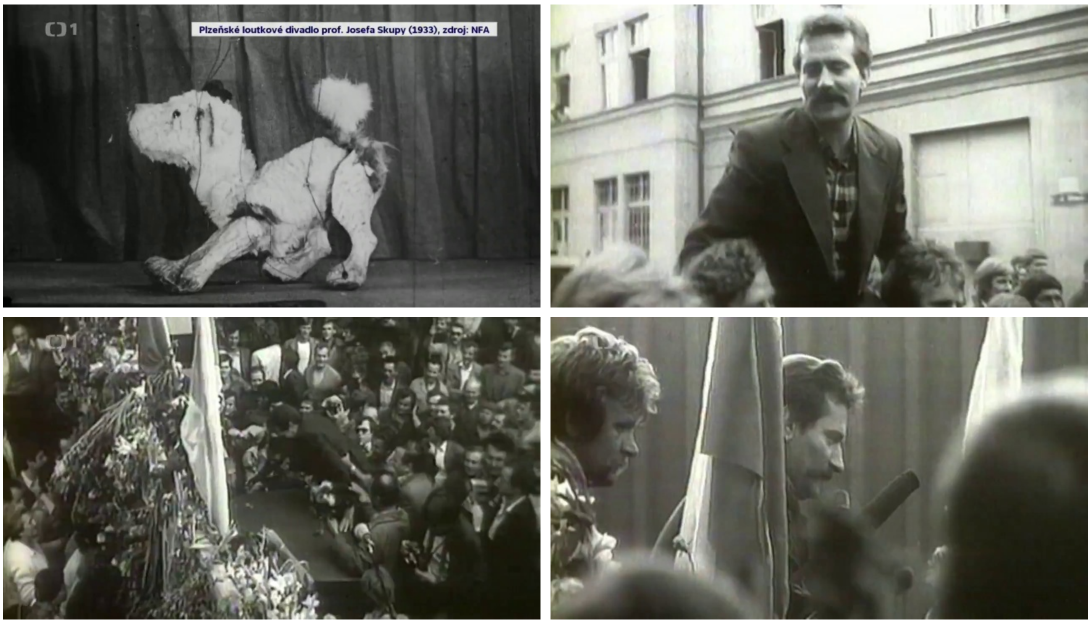

Plzeňské loutkové divadlo prof. Josefa Skupy (1933), zdroj: NFA

Figure A.2: Examples for Historic class

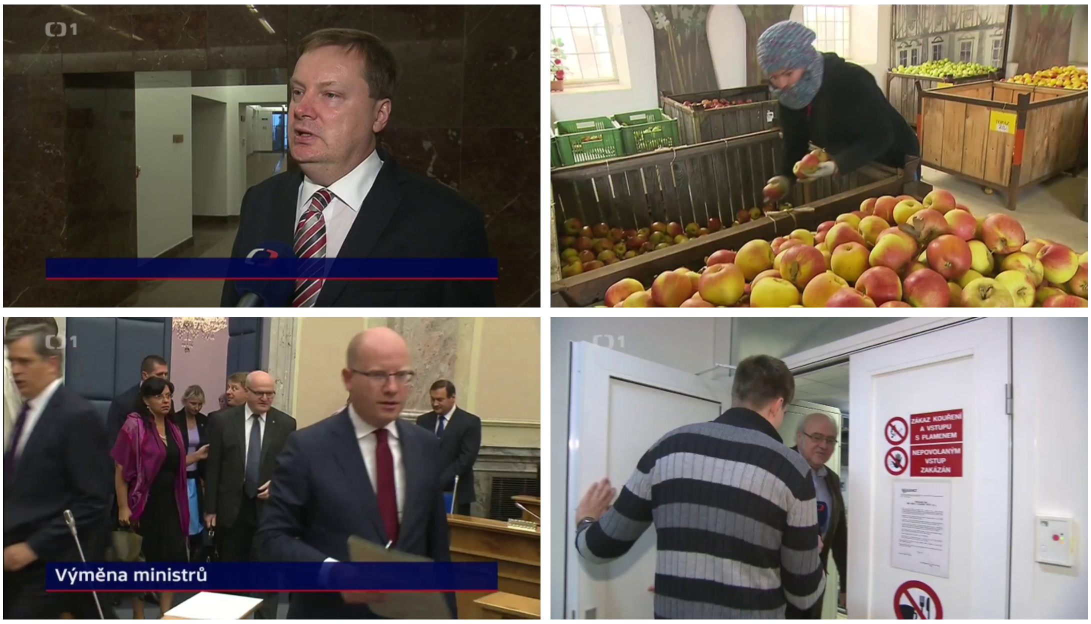


Figure A.3: Examples for Indoor class

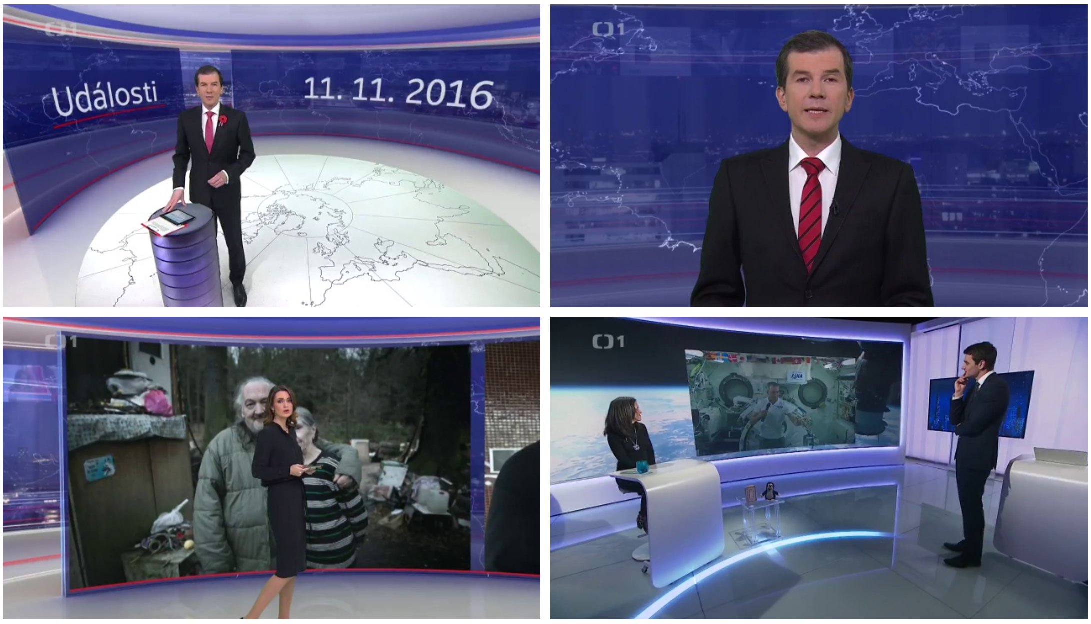


Figure A.4: Examples for Studio class

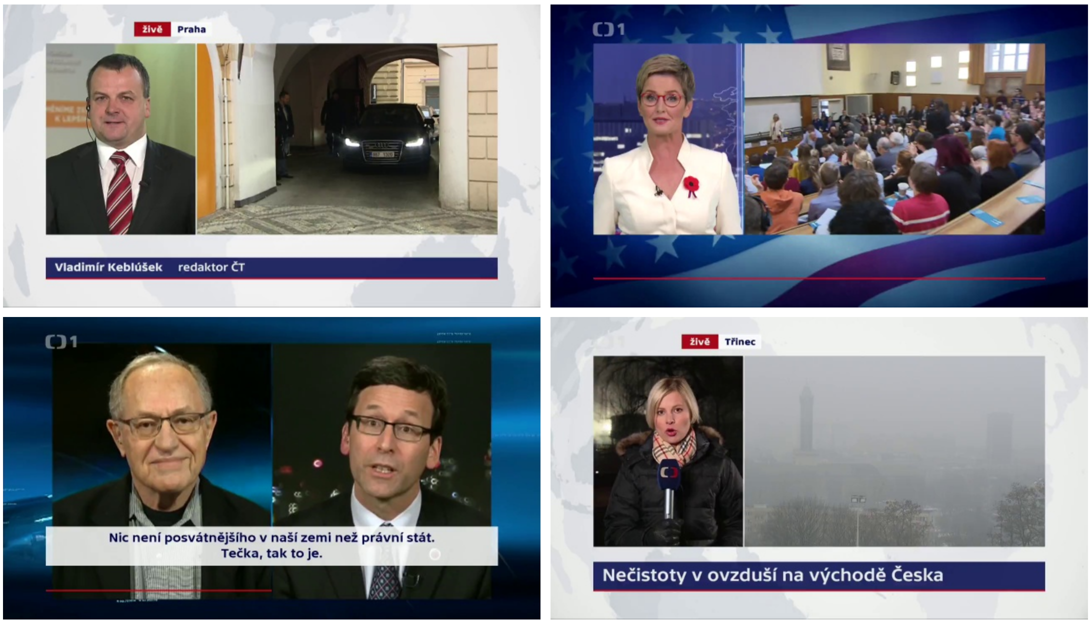

živě Praha
Vladimír Keblúšek redaktor ČT
Nic není posvátnějšího v naší zemi než právní stát.
Tečka, tak to je.
živě Třinec
Nečistoty v ovzduší na východě Česka

Figure A.5: Examples for Mix class

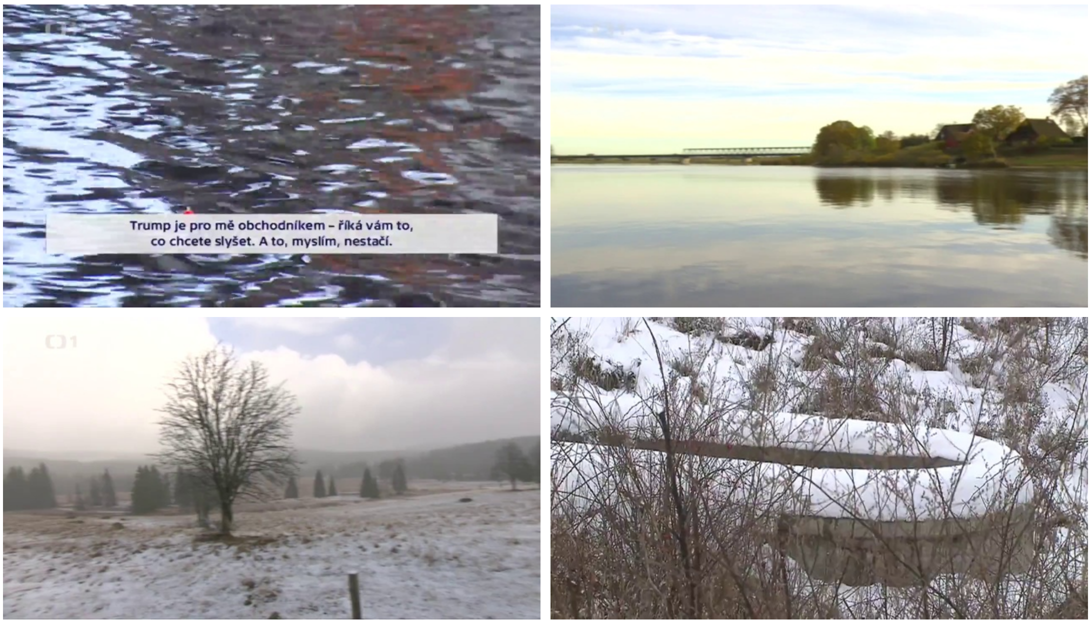

Trump je pro mě obchodníkem – říká vám to,
co chcete slyšet. A to, myslím, nestačí.

Figure A.6: Examples for Outdoor country class

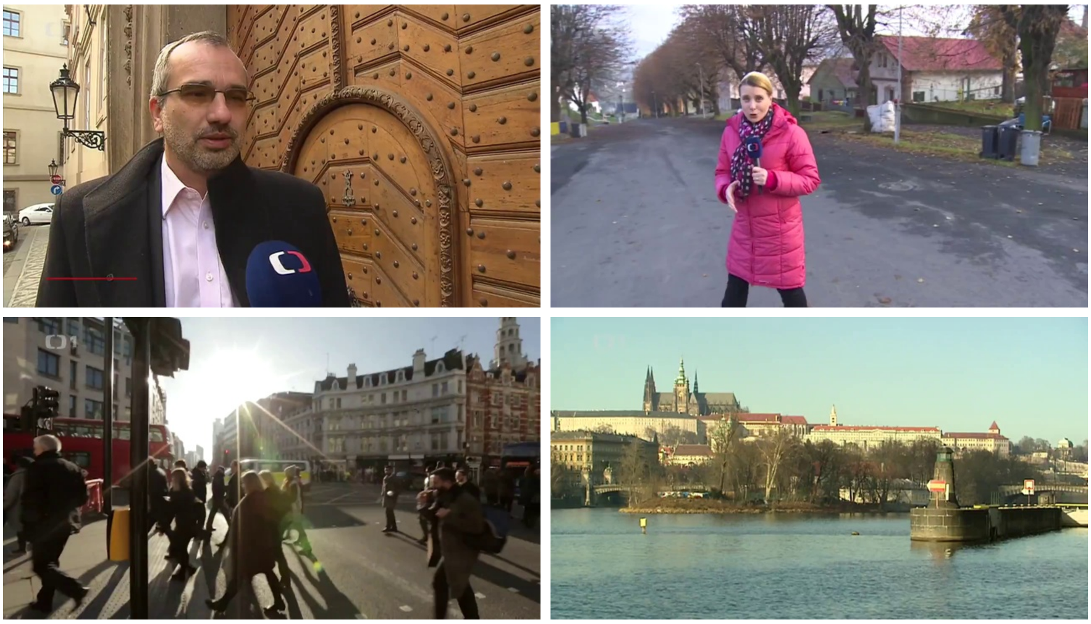


Figure A.7: Examples for Outdoor human-made class

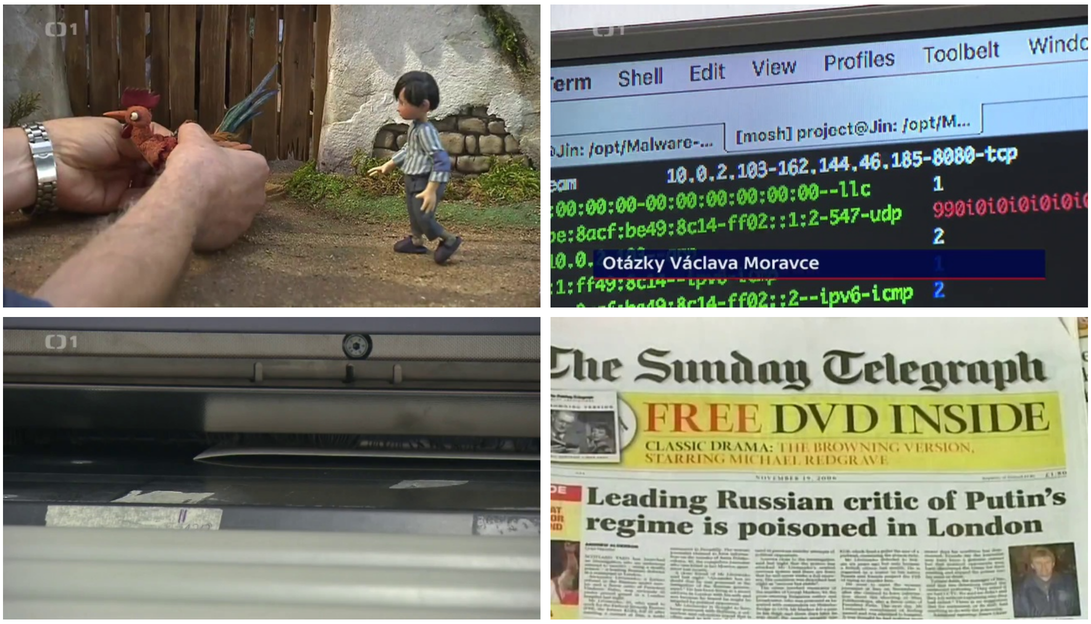

Term Shell Edit View Profiles Toolbelt Windo
Jin: /opt/Malware-...
[mosh] project@Jin: /opt/M...
10.0.2.103-162.144.46.185-8080-tcp
00:00:00-00:00:00:00:00:00--llc
e:8acf:be49:8c14-ff02::1:2-547-udp
Otázky Václava Moravce
The Sunday Telegraph
FREE DVD INSIDE
CLASSIC DRAMA: THE BROWNING VERSION, STARRING MICHAEL REDGRAVE
Leading Russian critic of Putin's regime is poisoned in London

Figure A.8: Examples for Other class

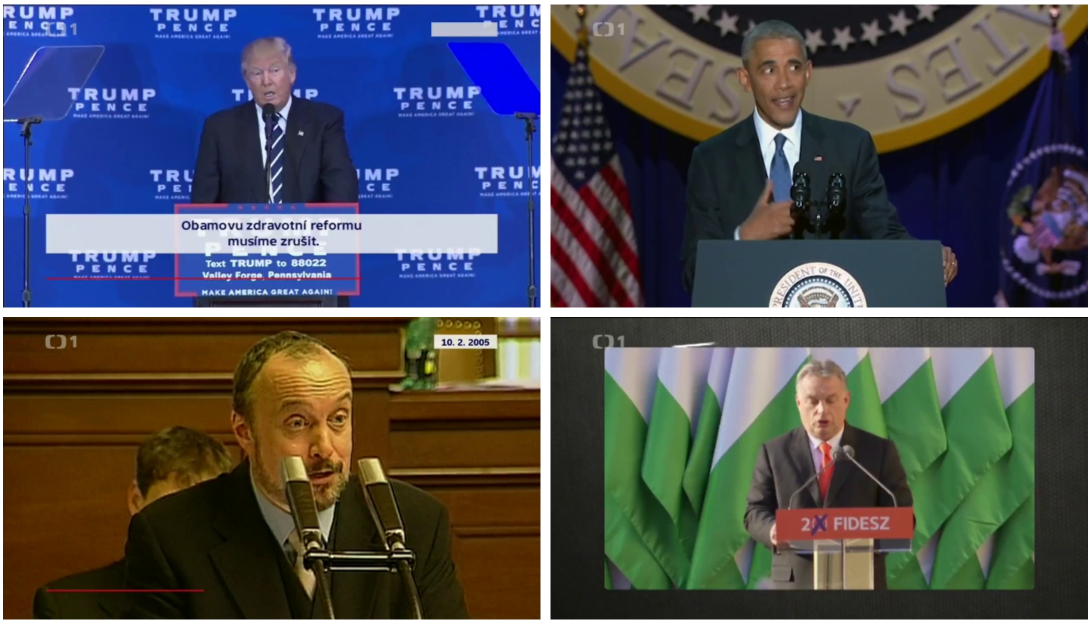

TRUMP PENCE
Obamovu zdravotní reformu musíme zrušit.
Text TRUMP to 88022
Valley Forge, Pennsylvania
MAKE AMERICA GREAT AGAIN!
10. 2. 2005
FIDESZ

Figure A.9: Examples for Speech class

A.2 Examples from television Prima

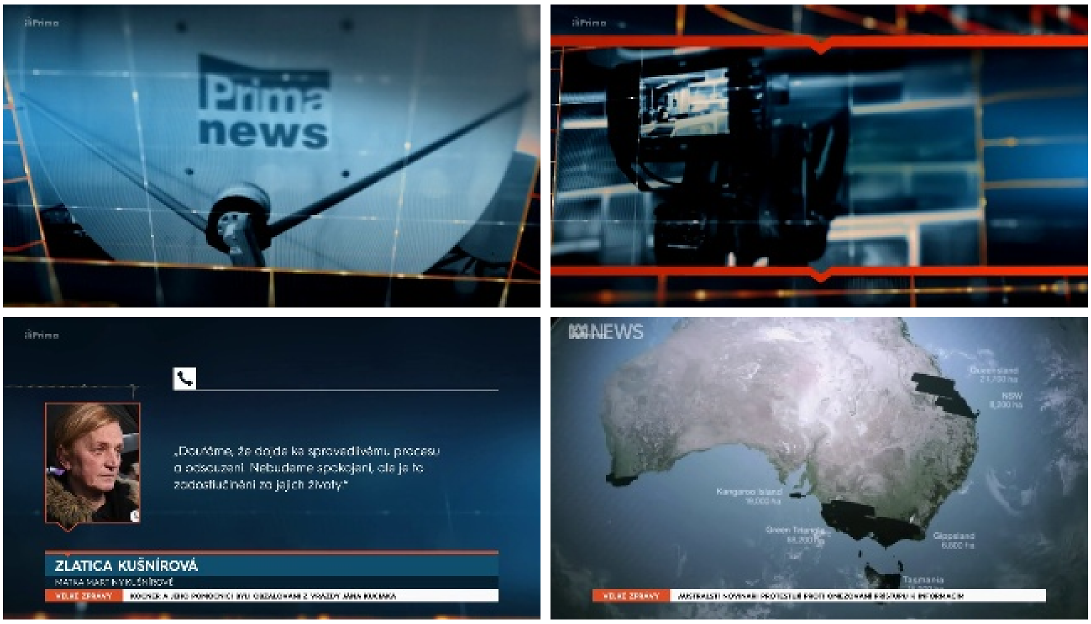


Figure A.10: Examples for Graphics class

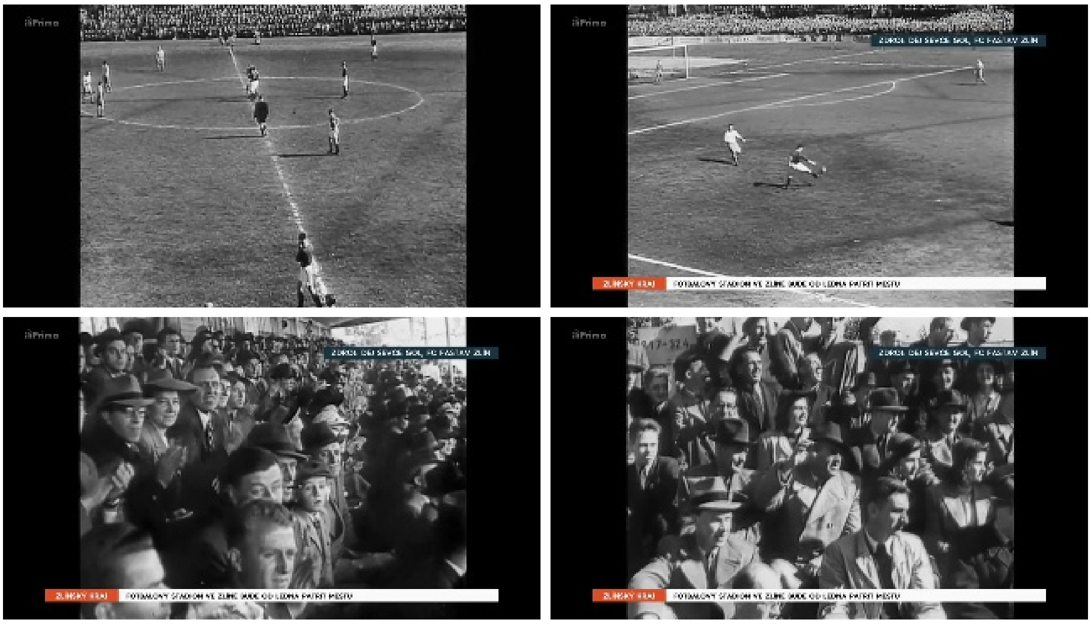


Figure A.11: Examples for Historic class

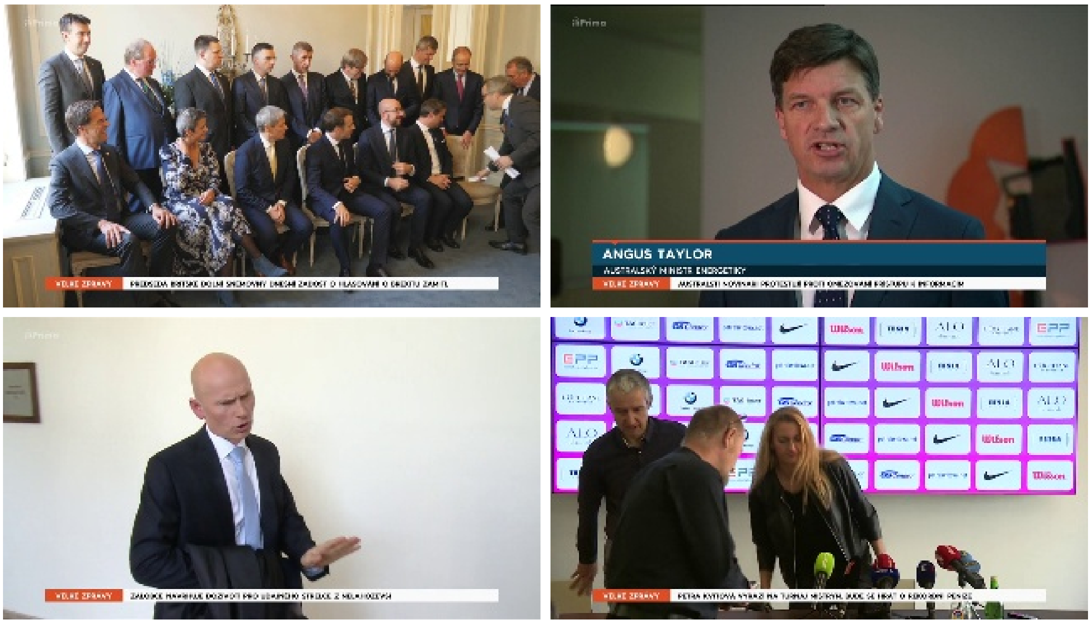


Figure A.12: Examples for Indoor class

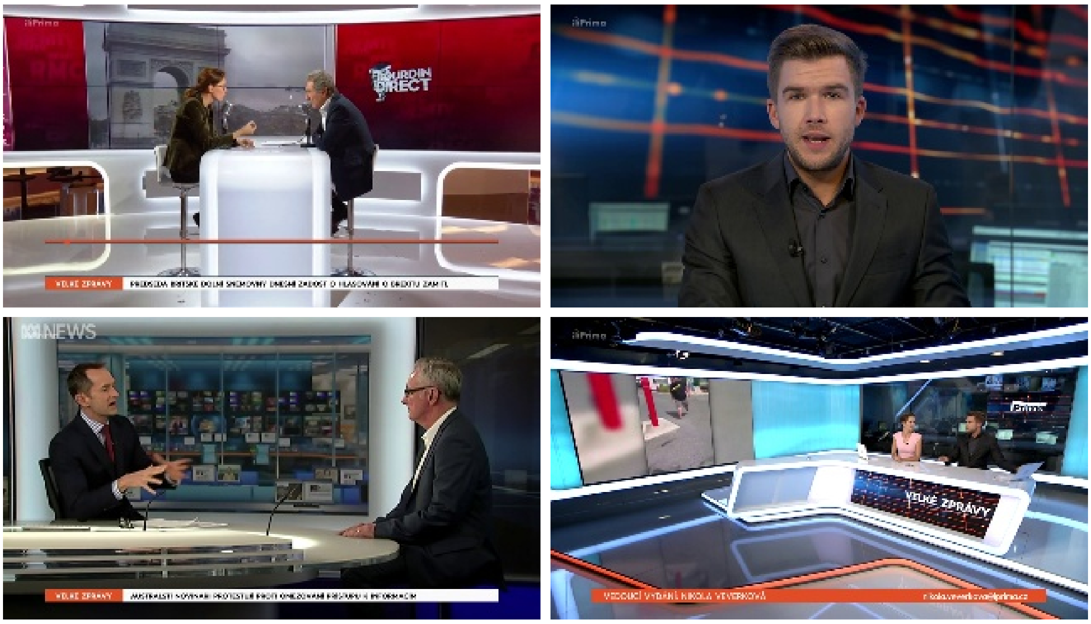


Figure A.13: Examples for Studio class

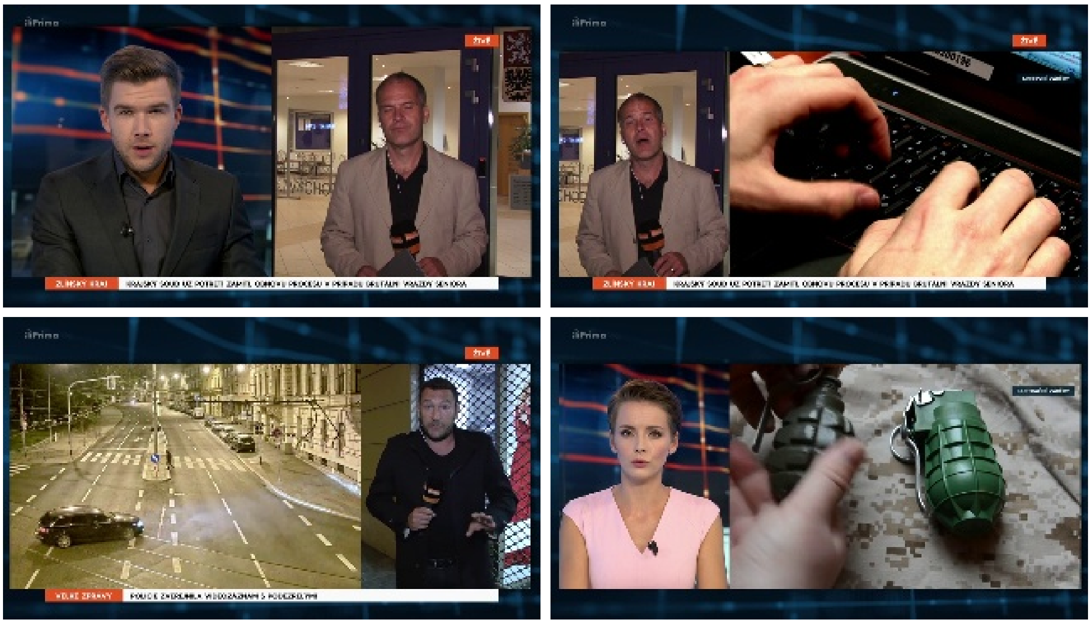


Figure A.14: Examples for Mix class

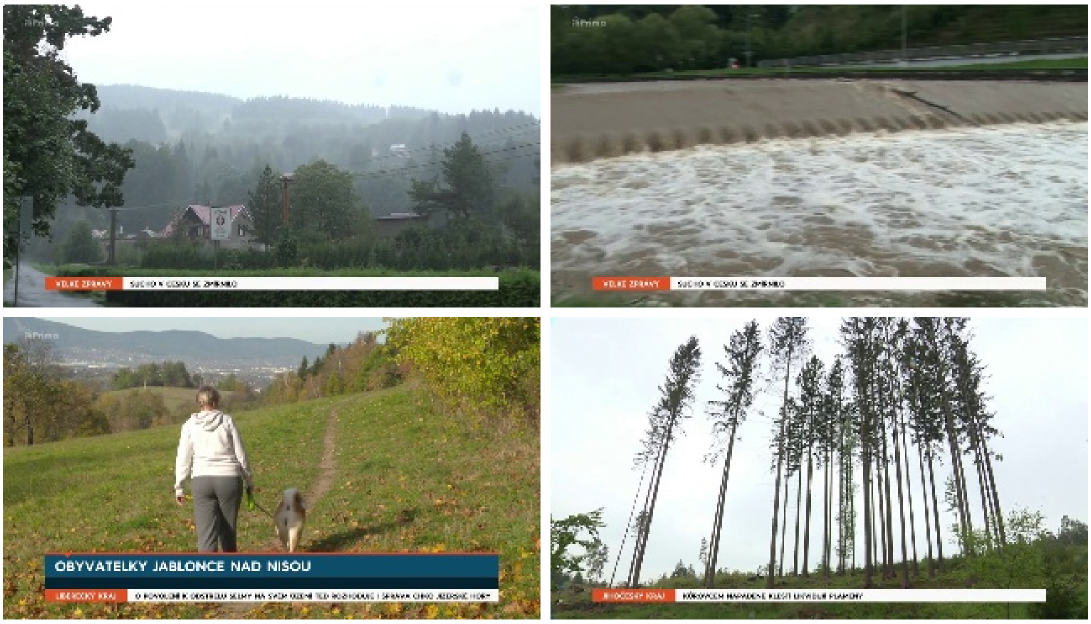


Figure A.15: Examples for Outdoor country class

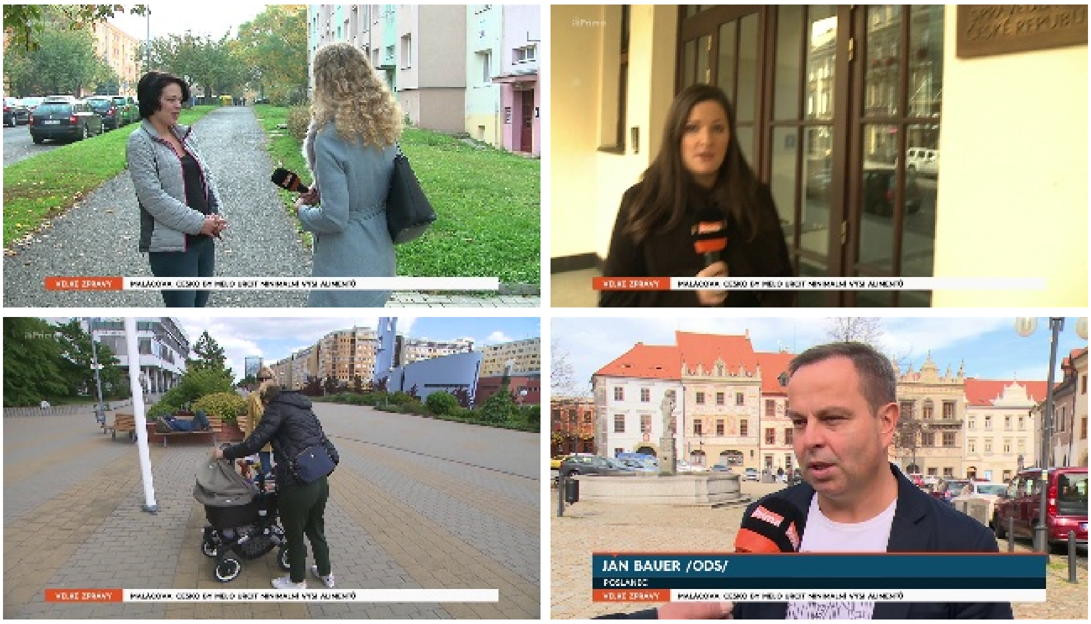


Figure A.16: Examples for Outdoor human-made class

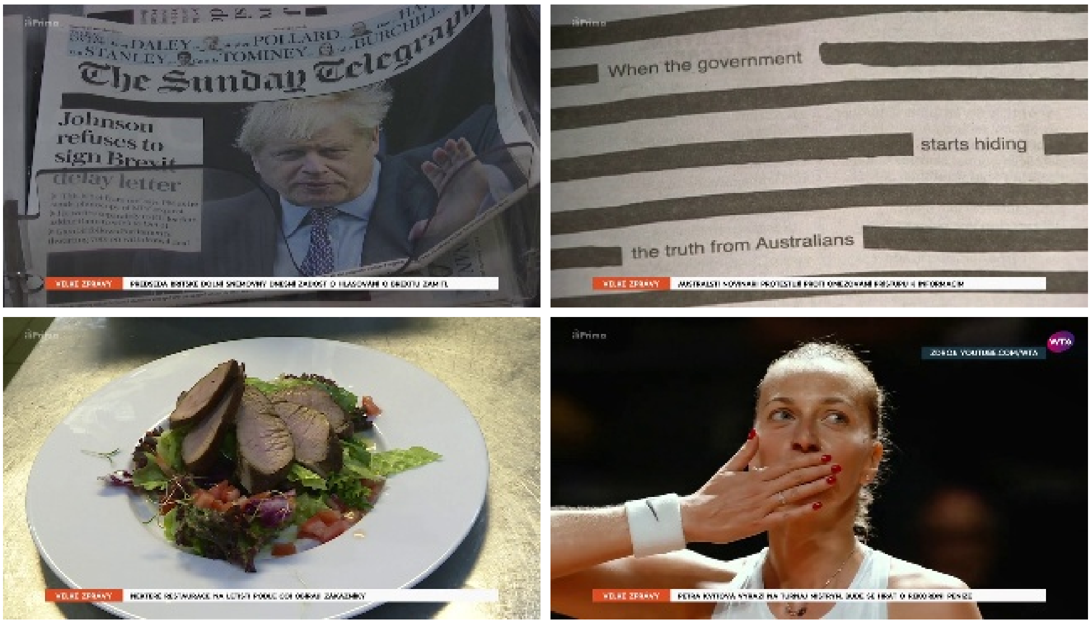

The Sunday Telegraph
Johnson refuses to sign Brexit delay letter
When the government
starts hiding
the truth from Australians

Figure A.17: Examples for Other class

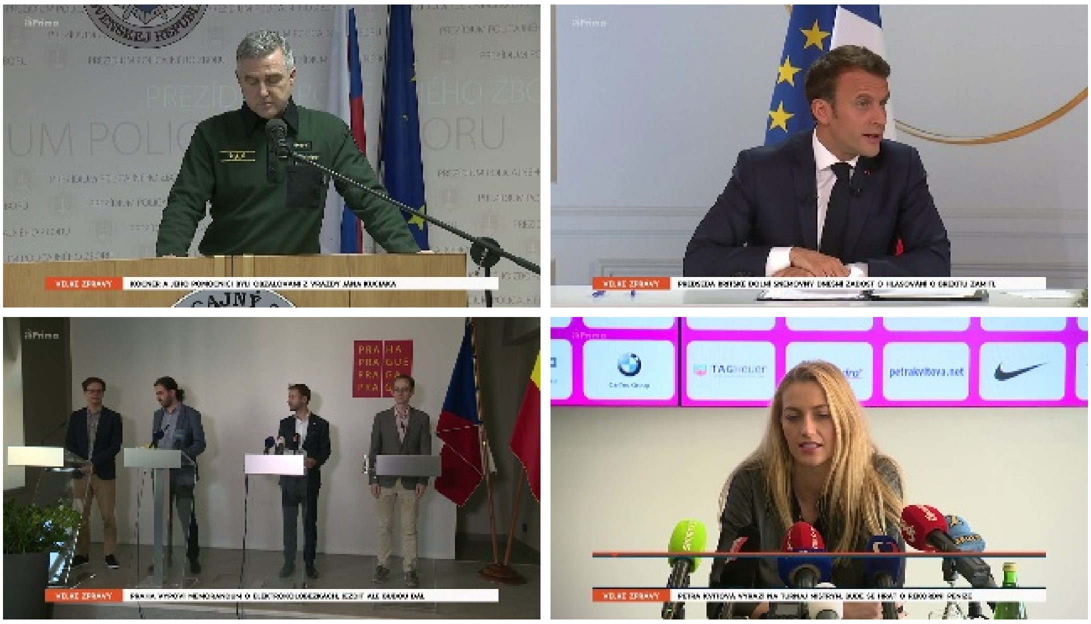

VELKÉ ZPRÁVY
KOČNER A JEHO POMOCNÍCI BYLI OBŽALOVÁNI Z VRAŽDY JÁNA KUCIAKA
VELKÉ ZPRÁVY
VELKÉ ZPRÁVY
PRAHA VYPOVÍ MEMORANDUM O ELEKTRONOLOGIZKÁCH, IZZIT ALE BUDOU DÁL
PRA HA
PRA GUE
PRA GA
PRA
VELKÉ ZPRÁVY
PETRA KVITOVÁ VYRAZÍ NA TURNAJ MISTRYŇ, BUDE SE HRÁT O REKORDNÍ PENÍZE
petrakvitova.net
TAGHeuer

Figure A.18: Examples for Speech class

Appendix B

Trained models

B.1 Various number of neurons in the fully-connected layer

Epochs	Number of neurons in the fully-connected layer							
	1,024	1,280	1,536	1,792	2,048	2,304	2,560	2,816
1	0.787	0.778	0.793	0.802	0.789	0.809	0.776	0.816
2	0.798	0.816	0.796	0.802	0.817	0.805	0.787	0.803
3	0.801	0.793	0.811	0.826	0.813	0.811	0.815	0.793
4	0.790	0.804	0.808	0.836	0.798	0.812	0.803	0.782
5	0.801	0.802	0.789	0.816	0.804	0.809	0.792	0.809
6	0.809	0.791	0.791	0.800	0.793	0.817	0.800	0.817
7	0.814	0.807	0.784	0.814	0.802	0.819	0.801	0.767
8	0.796	0.799	0.787	0.790	0.763	0.817	0.795	0.782
9	0.830	0.796	0.804	0.822	0.814	0.801	0.805	0.804
10	0.818	0.792	0.803	0.791	0.800	0.800	0.799	0.779
11	0.821	0.808	0.810	0.799	0.813	0.824	0.800	0.803
12	0.827	0.817	0.809	0.819	0.813	0.813	0.817	0.794
13	0.809	0.809	0.807	0.820	0.798	0.821	0.818	0.809
14	0.818	0.805	0.807	0.821	0.820	0.820	0.817	0.809
15	0.821	0.807	0.804	0.824	0.822	0.820	0.816	0.813
16	0.820	0.812	0.804	0.824	0.822	0.833	0.819	0.807
17	0.820	0.801	0.782	0.816	0.817	0.821	0.818	0.818
18	0.826	0.812	0.808	0.815	0.821	0.827	0.823	0.819
19	0.819	0.805	0.805	0.829	0.814	0.810	0.821	0.816
20	0.807	0.815	0.811	**0.830**	0.816	0.818	0.822	0.821
$\bar{x}_{[13,20]}$	0.818	0.808	0.803	**0.822**	0.816	0.821	0.819	0.814

Table B.1: Accuracies of individual epochs and average accuracy (denoted as $\bar{x}$) of the last eight epochs for given validation dataset using various amounts of neurons in the fully-connected layer. The subscript indicates the boundary index of epochs from which the average was calculated.

B.2 Backbone architectures

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.802	0.010	0.073	0.000	0.000	0.021	0.073	0.021	0.000
	Historic	0.075	0.509	0.000	0.000	0.000	0.321	0.075	0.019	0.000
	Indoor	0.003	0.003	0.917	0.000	0.001	0.002	0.058	0.012	0.004
	Studio	0.000	0.000	0.009	0.982	0.000	0.000	0.009	0.000	0.000
	Mix	0.212	0.000	0.019	0.000	0.769	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.007	0.060	0.000	0.000	0.497	0.423	0.013	0.000
	Outdoor human-made	0.008	0.016	0.172	0.000	0.000	0.046	0.753	0.004	0.000
	Other	0.032	0.016	0.397	0.000	0.000	0.095	0.365	0.095	0.000
	Speech	0.000	0.000	0.591	0.000	0.000	0.000	0.000	0.000	0.409

Table B.2: Normalized confusion matrix over the validation dataset of the VGG16 network using a Flatten layer followed by a classification one.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.771	0.010	0.062	0.000	0.000	0.042	0.104	0.010	0.000
	Historic	0.340	0.245	0.019	0.000	0.000	0.132	0.264	0.000	0.000
	Indoor	0.003	0.004	0.934	0.001	0.001	0.002	0.046	0.007	0.003
	Studio	0.000	0.000	0.005	0.991	0.005	0.000	0.000	0.000	0.000
	Mix	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.000	0.040	0.000	0.000	0.517	0.443	0.000	0.000
	Outdoor human-made	0.006	0.002	0.165	0.000	0.000	0.042	0.780	0.002	0.003
	Other	0.032	0.000	0.556	0.000	0.000	0.063	0.317	0.016	0.016
	Speech	0.045	0.000	0.591	0.000	0.000	0.000	0.000	0.000	0.364

Table B.3: Normalized confusion matrix over the validation dataset of the VGG16 network using a Flatten layer followed by fully-connected and classification layers.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.740	0.010	0.062	0.000	0.000	0.073	0.104	0.010	0.000
	Historic	0.264	0.189	0.019	0.000	0.000	0.151	0.302	0.075	0.000
	Indoor	0.004	0.004	0.932	0.000	0.000	0.005	0.047	0.008	0.001
	Studio	0.000	0.000	0.005	0.991	0.000	0.000	0.005	0.000	0.000
	Mix	0.000	0.000	0.019	0.019	0.962	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.000	0.027	0.000	0.000	0.409	0.550	0.013	0.000
	Outdoor human-made	0.001	0.000	0.172	0.000	0.000	0.037	0.788	0.000	0.002
	Other	0.016	0.016	0.460	0.000	0.000	0.048	0.444	0.016	0.000
	Speech	0.000	0.045	0.545	0.000	0.000	0.000	0.000	0.000	0.409

Table B.4: Normalized confusion matrix over the validation dataset of the InceptionV3 network using a Flatten layer followed by a classification one.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.729	0.010	0.052	0.000	0.000	0.052	0.125	0.031	0.000
	Historic	0.340	0.377	0.000	0.000	0.000	0.151	0.094	0.038	0.000
	Indoor	0.003	0.003	0.898	0.000	0.001	0.011	0.073	0.006	0.006
	Studio	0.000	0.000	0.009	0.991	0.000	0.000	0.000	0.000	0.000
	Mix	0.000	0.000	0.019	0.000	0.981	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.007	0.067	0.000	0.000	0.510	0.416	0.000	0.000
	Outdoor human-made	0.006	0.001	0.130	0.000	0.000	0.039	0.814	0.007	0.002
	Other	0.016	0.016	0.397	0.000	0.000	0.079	0.413	0.079	0.000
	Speech	0.000	0.045	0.545	0.000	0.000	0.000	0.000	0.000	0.409

Table B.5: Normalized confusion matrix over the validation dataset of the InceptionV3 network using a Flatten layer followed by fully-connected and classification layers.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.760	0.010	0.062	0.000	0.010	0.073	0.073	0.010	0.000
	Historic	0.094	0.415	0.019	0.000	0.000	0.132	0.075	0.264	0.000
	Indoor	0.001	0.002	0.910	0.000	0.000	0.006	0.071	0.007	0.003
	Studio	0.000	0.000	0.000	0.995	0.005	0.000	0.000	0.000	0.000
	Mix	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.000	0.034	0.000	0.000	0.691	0.275	0.000	0.000
	Outdoor human-made	0.000	0.004	0.092	0.000	0.000	0.055	0.835	0.012	0.002
	Other	0.032	0.000	0.381	0.000	0.000	0.048	0.460	0.063	0.016
	Speech	0.000	0.045	0.591	0.000	0.000	0.000	0.000	0.000	0.364

Table B.6: Normalized confusion matrix over the validation dataset of the InceptionResNetV2 network using a Flatten layer followed by a classification one.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.656	0.010	0.042	0.000	0.000	0.083	0.188	0.021	0.000
	Historic	0.208	0.377	0.000	0.000	0.000	0.151	0.113	0.151	0.000
	Indoor	0.002	0.003	0.908	0.000	0.001	0.011	0.069	0.001	0.006
	Studio	0.000	0.000	0.000	0.991	0.005	0.000	0.005	0.000	0.000
	Mix	0.000	0.000	0.019	0.000	0.981	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.000	0.020	0.000	0.000	0.678	0.302	0.000	0.000
	Outdoor human-made	0.000	0.001	0.113	0.000	0.000	0.059	0.808	0.016	0.003
	Other	0.032	0.000	0.317	0.000	0.000	0.079	0.492	0.079	0.000
	Speech	0.045	0.045	0.545	0.000	0.000	0.000	0.045	0.000	0.318

Table B.7: Normalized confusion matrix over the validation dataset of the InceptionResNetV2 network using a Flatten layer followed by fully-connected and classification layers.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.781	0.010	0.021	0.010	0.031	0.052	0.083	0.010	0.000
	Historic	0.038	0.623	0.000	0.019	0.000	0.132	0.038	0.151	0.000
	Indoor	0.004	0.003	0.850	0.007	0.000	0.011	0.085	0.023	0.018
	Studio	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
	Mix	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.000	0.020	0.000	0.000	0.685	0.282	0.013	0.000
	Outdoor human-made	0.000	0.000	0.088	0.002	0.000	0.058	0.834	0.014	0.004
	Other	0.000	0.016	0.254	0.016	0.000	0.111	0.413	0.143	0.048
	Speech	0.000	0.045	0.455	0.045	0.000	0.000	0.045	0.000	0.409

Table B.8: Normalized confusion matrix over the validation dataset of the InceptionResNetV2 network using a Global Average Pooling layer followed by a the classification one.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.781	0.010	0.021	0.000	0.021	0.073	0.073	0.021	0.000
	Historic	0.075	0.642	0.000	0.000	0.000	0.151	0.019	0.113	0.000
	Indoor	0.001	0.003	0.884	0.001	0.000	0.001	0.075	0.018	0.018
	Studio	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
	Mix	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.000	0.027	0.000	0.000	0.658	0.295	0.020	0.000
	Outdoor human-made	0.000	0.002	0.099	0.000	0.000	0.051	0.824	0.021	0.003
	Other	0.000	0.016	0.270	0.000	0.000	0.032	0.429	0.175	0.079
	Speech	0.000	0.045	0.545	0.000	0.000	0.000	0.000	0.000	0.409

Table B.9: Normalized confusion matrix over the validation dataset of the InceptionResNetV2 network using a Global Average Pooling layer followed by fully-connected and classification layers.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.719	0.000	0.031	0.000	0.031	0.094	0.104	0.021	0.000
	Historic	0.151	0.585	0.000	0.000	0.000	0.075	0.075	0.113	0.000
	Indoor	0.001	0.002	0.910	0.000	0.000	0.000	0.075	0.008	0.004
	Studio	0.000	0.000	0.005	0.991	0.000	0.000	0.005	0.000	0.000
	Mix	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.000	0.020	0.000	0.000	0.664	0.302	0.013	0.000
	Outdoor human-made	0.000	0.001	0.100	0.000	0.000	0.030	0.853	0.012	0.004
	Other	0.032	0.016	0.365	0.000	0.000	0.079	0.381	0.079	0.048
	Speech	0.045	0.045	0.682	0.000	0.000	0.000	0.000	0.000	0.227

Table B.10: Normalized confusion matrix over the validation dataset of the InceptionResNetV2 network using a Global Max Pooling layer followed by a classification one.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.740	0.010	0.052	0.000	0.000	0.094	0.062	0.042	0.000
	Historic	0.075	0.491	0.000	0.000	0.000	0.038	0.189	0.208	0.000
	Indoor	0.004	0.003	0.872	0.001	0.000	0.004	0.084	0.017	0.016
	Studio	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
	Mix	0.000	0.000	0.000	0.019	0.981	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.000	0.013	0.000	0.000	0.671	0.315	0.000	0.000
	Outdoor human-made	0.000	0.001	0.102	0.000	0.000	0.064	0.817	0.014	0.002
	Other	0.032	0.016	0.286	0.000	0.000	0.063	0.556	0.048	0.000
	Speech	0.000	0.045	0.591	0.000	0.000	0.000	0.000	0.000	0.364

Table B.11: Normalized confusion matrix over the validation dataset of the InceptionResNetV2 network using a Global Max Pooling layer followed by fully-connected and classification layers.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.625	0.000	0.083	0.000	0.010	0.125	0.156	0.000	0.000
	Historic	0.000	0.057	0.075	0.000	0.000	0.132	0.736	0.000	0.000
	Indoor	0.006	0.001	0.776	0.001	0.001	0.016	0.190	0.004	0.004
	Studio	0.000	0.000	0.005	0.995	0.000	0.000	0.000	0.000	0.000
	Mix	0.000	0.000	0.019	0.000	0.981	0.000	0.000	0.000	0.000
	Outdoor country	0.047	0.007	0.195	0.000	0.000	0.342	0.403	0.007	0.000
	Outdoor human-made	0.003	0.001	0.265	0.000	0.000	0.098	0.630	0.002	0.001
	Other	0.000	0.000	0.524	0.000	0.000	0.063	0.413	0.000	0.000
	Speech	0.000	0.000	0.727	0.000	0.000	0.000	0.000	0.000	0.273

Table B.12: Normalized confusion matrix over the validation dataset of the MobileNetV2 network using a Flatten layer followed by a classification one.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.594	0.000	0.156	0.000	0.000	0.042	0.167	0.042	0.000
	Historic	0.000	0.000	0.302	0.000	0.000	0.019	0.679	0.000	0.000
	Indoor	0.004	0.002	0.775	0.000	0.000	0.011	0.198	0.002	0.007
	Studio	0.000	0.000	0.023	0.964	0.000	0.000	0.009	0.000	0.005
	Mix	0.000	0.000	0.000	0.000	0.981	0.000	0.000	0.000	0.019
	Outdoor country	0.060	0.000	0.228	0.000	0.000	0.275	0.396	0.034	0.007
	Outdoor human-made	0.009	0.006	0.392	0.001	0.001	0.068	0.512	0.007	0.002
	Other	0.000	0.000	0.603	0.016	0.000	0.016	0.317	0.000	0.048
	Speech	0.000	0.000	0.500	0.000	0.000	0.000	0.136	0.000	0.364

Table B.13: Normalized confusion matrix over the validation dataset of the MobileNetV2 network using a Flatten layer followed by fully-connected and classification layers.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.833	0.010	0.031	0.000	0.000	0.031	0.042	0.052	0.000
	Historic	0.302	0.547	0.000	0.000	0.000	0.019	0.075	0.057	0.000
	Indoor	0.001	0.006	0.903	0.001	0.000	0.006	0.056	0.020	0.006
	Studio	0.000	0.000	0.005	0.995	0.000	0.000	0.000	0.000	0.000
	Mix	0.038	0.000	0.000	0.000	0.962	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.000	0.020	0.000	0.000	0.718	0.262	0.000	0.000
	Outdoor human-made	0.001	0.002	0.128	0.000	0.000	0.055	0.806	0.007	0.001
	Other	0.048	0.000	0.317	0.000	0.000	0.063	0.460	0.111	0.000
	Speech	0.000	0.000	0.500	0.000	0.000	0.000	0.000	0.000	0.500

Table B.14: Normalized confusion matrix over the validation dataset of the MobileNetV2 network using a Global Average Pooling layer followed by a classification one.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.865	0.000	0.000	0.000	0.000	0.042	0.052	0.042	0.000
	Historic	0.000	0.679	0.000	0.000	0.000	0.189	0.038	0.094	0.000
	Indoor	0.001	0.007	0.907	0.000	0.000	0.006	0.045	0.020	0.013
	Studio	0.000	0.000	0.005	0.995	0.000	0.000	0.000	0.000	0.000
	Mix	0.000	0.000	0.000	0.000	0.981	0.000	0.000	0.000	0.019
	Outdoor country	0.000	0.000	0.034	0.000	0.000	0.678	0.282	0.007	0.000
	Outdoor human-made	0.002	0.004	0.115	0.000	0.000	0.069	0.772	0.034	0.003
	Other	0.000	0.048	0.254	0.000	0.000	0.063	0.429	0.206	0.000
	Speech	0.045	0.045	0.364	0.000	0.000	0.000	0.045	0.045	0.455

Table B.15: Normalized confusion matrix over the validation dataset of the MobileNetV2 network using a Global Average Pooling layer followed by fully-connected and classification layers.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.625	0.021	0.219	0.000	0.010	0.083	0.021	0.021	0.000
	Historic	0.340	0.075	0.151	0.000	0.000	0.038	0.340	0.057	0.000
	Indoor	0.011	0.023	0.910	0.001	0.000	0.008	0.037	0.009	0.001
	Studio	0.000	0.000	0.009	0.991	0.000	0.000	0.000	0.000	0.000
	Mix	0.019	0.000	0.019	0.000	0.962	0.000	0.000	0.000	0.000
	Outdoor country	0.007	0.000	0.396	0.000	0.000	0.342	0.228	0.027	0.000
	Outdoor human-made	0.010	0.014	0.367	0.000	0.000	0.152	0.431	0.023	0.002
	Other	0.032	0.000	0.762	0.000	0.000	0.079	0.032	0.063	0.032
	Speech	0.000	0.000	0.682	0.000	0.000	0.000	0.000	0.000	0.318

Table B.16: Normalized confusion matrix over the validation dataset of the MobileNetV2 network using a Global Max Pooling layer followed by a classification one.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	Historic	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	Indoor	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	Studio	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	Mix	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	Outdoor country	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	Outdoor human-made	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	Other	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	Speech	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000

Table B.17: Normalized confusion matrix over the validation dataset of the MobileNetV2 network using a Global Max Pooling layer followed by fully-connected and classification layers.

B.3 Time-distributed networks

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.891	0.008	0.050	0.008	0.008	0.008	0.017	0.008	0.000
	Historic	0.000	0.714	0.000	0.000	0.286	0.000	0.000	0.000	0.000
	Indoor	0.013	0.017	0.870	0.021	0.013	0.000	0.034	0.017	0.017
	Studio	0.000	0.000	0.000	0.993	0.007	0.000	0.000	0.000	0.000
	Mix	0.028	0.000	0.000	0.000	0.972	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.000	0.045	0.000	0.000	0.773	0.182	0.000	0.000
	Outdoor human-made	0.016	0.000	0.036	0.000	0.000	0.056	0.880	0.004	0.008
	Other	0.111	0.111	0.222	0.000	0.000	0.000	0.111	0.222	0.222
	Speech	0.000	0.000	0.111	0.000	0.000	0.000	0.000	0.000	0.889

Table B.18: Normalized confusion matrix over the validation dataset of the InceptionResNetV2 network using a Flatten layer followed by LSTM and classification layers.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.884	0.017	0.017	0.017	0.050	0.000	0.008	0.000	0.008
	Historic	0.286	0.714	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Indoor	0.031	0.000	0.774	0.071	0.022	0.000	0.058	0.004	0.040
	Studio	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
	Mix	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.000	0.000	0.000	0.000	0.909	0.000	0.000	0.091
	Outdoor human-made	0.012	0.000	0.004	0.024	0.020	0.053	0.878	0.004	0.004
	Other	0.158	0.211	0.158	0.000	0.000	0.000	0.158	0.263	0.053
	Speech	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000

Table B.19: Normalized confusion matrix over the validation dataset of the InceptionResNetV2 network using a Global Average Pooling layer followed by LSTM and classification layers.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.909	0.008	0.017	0.017	0.041	0.000	0.008	0.000	0.000
	Historic	0.000	0.714	0.000	0.143	0.143	0.000	0.000	0.000	0.000
	Indoor	0.000	0.000	0.898	0.044	0.004	0.000	0.013	0.009	0.031
	Studio	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
	Mix	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.000	0.000	0.000	0.000	0.909	0.091	0.000	0.000
	Outdoor human-made	0.008	0.000	0.008	0.016	0.004	0.045	0.894	0.000	0.024
	Other	0.158	0.158	0.263	0.000	0.000	0.105	0.105	0.211	0.000
	Speech	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000

Table B.20: Normalized confusion matrix over the validation dataset of the InceptionResNetV2 network using a Global Max Pooling layer followed by LSTM and classification layers.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.959	0.017	0.008	0.000	0.017	0.000	0.000	0.000	0.000
	Historic	0.286	0.714	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Indoor	0.027	0.009	0.814	0.053	0.018	0.000	0.022	0.013	0.044
	Studio	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
	Mix	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000
	Outdoor human-made	0.016	0.000	0.024	0.057	0.004	0.029	0.865	0.004	0.000
	Other	0.053	0.158	0.158	0.000	0.000	0.000	0.211	0.421	0.000
	Speech	0.000	0.000	0.000	0.000	0.000	0.000	0.091	0.000	0.909

Table B.21: Normalized confusion matrix over the validation dataset of the MobileNetV2 network using a Global Average Pooling layer followed by LSTM and classification layers.

		Predicted label								
		Graphics	Historic	Indoor	Studio	Mix	Outdoor country	Outdoor human-made	Other	Speech
True label	Graphics	0.917	0.017	0.033	0.000	0.017	0.000	0.008	0.008	0.000
	Historic	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Indoor	0.000	0.013	0.934	0.000	0.004	0.000	0.013	0.009	0.027
	Studio	0.000	0.000	0.004	0.996	0.000	0.000	0.000	0.000	0.000
	Mix	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
	Outdoor country	0.000	0.000	0.000	0.000	0.000	0.955	0.000	0.045	0.000
	Outdoor human-made	0.004	0.004	0.024	0.000	0.000	0.008	0.943	0.012	0.004
	Other	0.000	0.211	0.263	0.000	0.000	0.105	0.105	0.316	0.000
	Speech	0.000	0.000	0.091	0.000	0.000	0.000	0.000	0.000	0.909

Table B.22: Normalized confusion matrix over the validation dataset of the MobileNetV2 network using a Global Average Pooling layer followed by fully-connected, LSTM, and classification layers.