Laboratory of
Artificial Intelligence

Artificial Intelligence &
Machine Learning Group

Master Thesis

Deep Models and Transfer Learning for Facial Emotion Recognition

Georgios Kyritsis

Supervisors

EPFL: Prof. Boi Faltings
Swisscom: Dr. Claudiu Musat

March 16, 2018

Abstract

Facial emotion recognition which is a challenging task has attracted research interest within the field of artificial intelligence. Many approaches are based on engineered features (e.g HOG, SIFT, Gabor Wavelets, and LPQ) where these features are given as input to classifiers. Nevertheless, the results generalise poorly to previously unseen data.

Recently, deep convolutional neural networks (CNN) have been trained to achieve state-of-the-art performance on a wide variety of tasks including, natural language processing, image classification, and speech recognition. This paper proposes a deep neural network architecture to address the FER problem which in conjunction with facial action units achieves state-of-the-art performance. Facial action unit is one of the most powerful and immediate means for humans to communicate their emotions.

The input images for our network are from the FER+ dataset which provides one emotion label for each face. Our network classifies each face based on the emotion shown in the facial expression in one of the eight categories (neutral, happy, surprise, sad, angry, disgust, fear, and contempt).

Acknowledgements

I would like to express my deep gratitude to my advisor at EPFL, Prof. Boi Faltings for giving me the opportunity to work on such an interesting research topic.

Also I would like to express my very great appreciation to Dr. Claudiu Musat for his valuable and constructive recommendations and critiques during the development of this research work. His willingness to give his time has been very much appreciated.

Also I would like to extend my thanks to my fellow teammates for all the constructive and encouraging discussions that we had during the semester.

Finally, I would like to thank my parents and my brother for their support and encouragement throughout my Master studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Identifying human emotion most typically from facial expressions has become an increasingly important research area. It involves signal processing, machine learning, and computer vision and can be used in many applications such as human-computer interaction [41], driver safety [42], and health care [25].

Many inputs can be used to recognize emotions. Among them facial expression is the most popular. In a paper published on 1971 Paul Ekman et al. [13] identified 6 emotions that are universal across different cultures: happiness, surprise, sadness, anger, disgust, and fear. Moreover, Ekman developed the Facial Action Coding System (FACS) [14] which is a tool for measuring facial expressions. It breaks down facial expressions into individual components of muscle movement using standard facial substructures called Action Units. Each Action Unit is based on one or few facial muscles. This system became for many years the standard tool for emotion recognition. However, FACS requires experts to annotate images which is both expensive and time consuming. So it is hard to build an emotion recognition system that is mainly based on this tool.

Artificial intelligence has been used in many different domains including computer vision, speech recognition, natural language processing, where they achieved results comparable or in some cases superior to humans. Neural networks and deep learning currently provide the best solutions to many of the above mentioned problems. Neural network is a type of graph that takes an input, applies a function at each node also called neuron and outputs a scalar value. Each neuron is connected to every neuron from the previous layer and every neuron operates separately from the neurons that belong to the same layer. Each edge also called connection is associated with a weight that represent the strength of connections between neurons. If the weight from neuron I to neuron II has greater magnitude, it means that neuron I has great influence over neuron II. Training a neural network refers to the task of determining the best set of weights for maximizing the neural network's accuracy. Figure 1.1 depicts a simple Neural Network with 2 hidden layers.

Deep convolutional neural networks (CNNs) which are very similar to ordinary Neural Networks that are discussed previously, have achieved breakthrough accuracies to image classification [24] that can easily be extended to the problem of facial expression recognition. Convolutional neural networks are made up of neurons that have learnable weights and biases. Each neuron receives an input performs a dot product and after that it performs a non-linear transformation. Deep CNNs are composed of several layers, and every layer transforms one volume of activations to

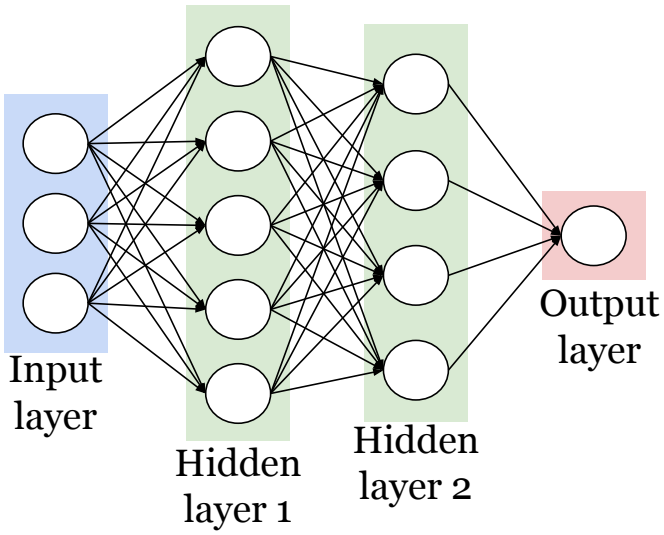

Figure 1.1: Neural Network Schematic

another through a non linear differential function. Recent state-of-the-art architectures have used a number of additional components to enhance and complement the convolutional operation. The major components are:

- Pooling layer
- ReLU layer [27]
- Dropout layer [37]
- Batch normalization [20]

It is desirable to periodically introduce pooling layers between convolutional layers in order to reduce the spatial size of the image. The pooling layer operates on each local region of an image and computes the average of the pixel values in the region (Average pooling) or keeps the maximum pixel value of the region (Max pooling). The idea behind pooling is that makes CNNs invariant to variations in the representation of the image and reduces the effect of background. Also pooling reduces the number of trainable parameters and as a results the training of the network becomes faster.

ReLU layer applies an elementwise activation function $\max(0, x)$, which turns negative values to zeros (thresholding at zero).

Dropout layer provides a simple way to avoid overfitting. Overfitting is a phenomenon in which a model works well on the training dataset, but performs poorly on new unseen data. This happens since the model has learned both the signal and the noise from the training dataset. The idea behind dropout layers is to randomly set some activations to 0, essentially dropping components from a layer of the neural network. By doing this, the network is forced to find more ways to classify the images instead of over-depending on some features.

The idea behind batch normalisation is that instead of only normalising the input of the network, we normalise the inputs to layers within the network. Batch

normalisation helps the network in many ways: networks train faster, allows higher learning rates, provides some regularisation, and makes weights easier to initialise.

Figure 1.2 depicts a simple convolutional neural network with convolutional layers, subsampling layers (pooling layers) followed by fully connected layers. We stack these three types of layers to form a Convolutional neural network architecture. The input is an image of dimensions $h \times w \times r$, where h is the height, w is the width, and r is the number of channels, e.g $r = 3$ for an RGB image and $r = 1$ for a grayscale image. The convolutional layer has k filters of size $m \times m \times d$, where m is the filter size and obviously has to be smaller than the input image, and d is equal to the depth of the input image, e.g for an RGB image $d = 3$. The output is k images of size $(h - m + 1) \times (w - m + 1)$. Usually an additive bias and and a non-linear activation function is applied to each image leaving the image size unchanged. So the input to the next convolutional layer will be an image of size $(h-m+1) \times (w-m+1) \times k$. As mentioned before between the convolutional layers we insert a pooling layer that reduces the spatial size of the representation. After this operation the depth remains unchanged while the width and the height decrease proportional to the size of pooling layer p. After these layers we have any number of fully connected layers that have full connections to all activations from the previous layer as is the case to layers in a standard multilayer neural network. The fully connected layer uses a softmax activation function in the output layer. Putting it all together the convolutional and pooling layers act as feature extractors from the input image. For example first layers will learn edges and second layers will combine these edges in order to form more abstract representations like rectangular shapes. Next layers will combine these layers to create more abstract representations like a human eye or nose. So far we have described a feed-forward network that can be thought as a composition of a number of functions:

$$f(\mathbf{x}; \mathbf{w}) = f_L(...f_2(f_1(\mathbf{x}; \mathbf{w_1}); \mathbf{w_2})...), \mathbf{w_L})$$

Each function takes as input a data vector $\mathbf{x}_l$ where l denotes the layer and and a parameter vector $\mathbf{w}_l$ and produces an output $\mathbf{x}_{l+1}$. The parameters $\mathbf{w} = (\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_L})$ for each layer l are learned from the training dataset. In the case of image classification the data vectors $\mathbf{x}$ are 3D arrays of pixels (height, width, channels). The parameters $\mathbf{w} = (\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_L})$ should be learned in such a manner that the overall function $f(\mathbf{x}; \mathbf{w})$ achieves a goal, that is minimising a loss function $L = l(\mathbf{z}, \hat{\mathbf{z}})$ that expresses the penalty for predicting $\hat{\mathbf{z}}$ instead of $\mathbf{z}$. Learning will be achieved by adjusting the weights $\mathbf{w}$ such that the predicting $\hat{\mathbf{z}}$ is as close as possible to $\mathbf{z}$. The simplest algorithm to minimize L is gradient descent[1]. The idea behind gradient descent is that you update the parameters $\mathbf{w}$ along the direction of the fastest descent of the loss function L. So the update of the parameters at every step t is given by the following formula:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta_t \frac{\partial f}{\partial \mathbf{w}}(\mathbf{w}^t)$$
where η_t is the learning rate at time step t

The method used to train CNNs and which is based on gradient descent is called backpropagation[2]. The idea is that the parameters of the models are adjusted in proportion to their contribution to the total error. So after this algorithm the

[1] `https://en.wikipedia.org/wiki/Gradient_descent`

[2] `https://en.wikipedia.org/wiki/Backpropagation`

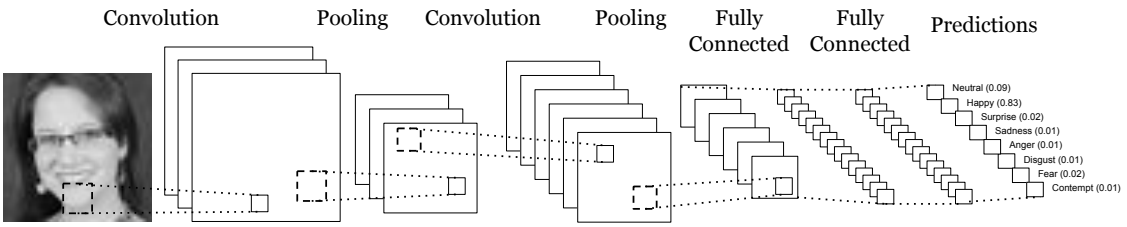

Figure 1.2: Convolutional Neural Network Schematic

architecture is able to classify correctly the images from the training dataset. Given that the training dataset is large enough, the architecture will generalise well to unseen images and classify them into correct categories. So in order to predict the category of a new image, the network will go through a forward propagation step with the optimal weights calculated by the backpropagation algorithm and will output a probability for each class. So the answer will be the category with the highest probability.

In this paper, we experiment with the latest deep convolutional neural networks in order to classify human faces into discrete emotion labels. We train from scratch architectures including the VGG [36], ResNet [16], GoogLeNet [38], DenseNet [19], and Wide ResNet [46] networks. Also we experiment on transfer learning and multi-task learning. Transfer learning is a machine learning method where a model trained for a task is used as a starting point for a different task, while in multi-task learning multiple learning tasks are solved at the same time. Multiple CNNs are available such as VGG16 that are pre-trained on ImageNet [7] or to perform facial recognition [30]. These pre-trained models are used as feature extractors. With these features we train a linear support vector machine. Both high and low level features are taken from the pre-trained networks. The rest of the paper is organized as follows. Related works are discussed in Section 2 and a description of the datasets used, in Section 2. Section 4 gives a brief description of the most popular CNN architectures, while sections 5 and 6 present two popular methods that are used in deep learning. Section 7 presents the experiments with their results, while section 8 explains in detail the algorithm that achieves state-of-the-art accuracy on the FER+ dataset. Finally, section 9 gives an overview of our analysis.

Chapter 2

Related Work

Facial emotion recognition has been an active research topic for decades. Early approaches were based on manually designed or hand-crafted features such as SIFT [4], Gabor Wavelets [48], Histograms of Oriented Gradients (HOGs) [9], Local Binary Patterns (LBP) [35], Local Binary Patterns on Three Orthogonal Planes (LBP-TOP) [49], Local Phase Quantization (LPQ) [43], and Action Units [40]. All these models lack generalization and their performance degrades significantly when they are applied to datasets that are different to the original dataset.

In recent years, convolutional neural networks (CNNs) have revolutionized computer vision. They were popularized by Yann LeCun's LeNet [24] at the 1990s but only recently it has been able to train these networks on large datasets due to the increased processing power afforded by graphical processing units (GPUs). We are now able to train very deep networks with millions of parameters on very large datasets like ImageNet [7]. Image classification, which is major problem in computer vision, has been successfully tackled by deep convolutional neural networks [23, 38]. Since emotion recognition is an image classification task, CNN is also applied to this problem. In particular, Yu and Zhang achieved state-of-the-art results for the Emotion Recognition in the Wild challenge (EmotiW) [10] in 2015 by using an ensemble of multiple deep convolutional neural networks with 5 convolutional layers each [45]. A novelty of this paper is the usage of stochastic pooling [47] instead of max pooling. Also they used data perturbation and a voting method that increased the accuracy of the model by 2-3%. Mollahoseini et al. [26] proposed an architecture consisting of 4 convolutional layers each followed by max pooling and then followed by 4 Inception layers as introduced by GoogLeNet [38]. The network was tested on many publicly available datasets, and achieved state-of-the-art accuracies on many of them. Barsoum et al. proposed a custom VGG13 model with 10 convolutional layers with max pooling and dropout layers in between that achieved state-of-the-art accuracy to the FER+ dataset [3].

Chapter 3

Datasets

In the facial expression recognition task, unlike other problems' datasets such as ImageNet [23], the available datasets are usually small and unbalanced, providing only a few examples to some emotion categories, making neural networks hard to train. Below we describe briefly the datasets that are used in our analysis.

3.1 FER+

The Facial Expression Recognition 2013 (FER2013) dataset, which was introduced in the ICML 2013 Challenges in Representation Learning [39], was created by Google image search API. The dataset consists of 48-by-48 pixel grayscale images of human faces annotated with the seven basic emotions (neutral, happiness, surprise, sadness, anger, disgust, and fear). The label accuracy of this dataset is not very high, and as a result a newer version of this dataset called FER+ has been released by Microsoft [3], where each image has been labeled by 10 crowd-sourced taggers, which provide better ground truth labels than the original FER labels. Figure 3.1 shows some examples of images with labels from the initial dataset(FER2013) and the newer version(FER+).

The FER+ dataset includes the label contempt to the list of possible emotions. Finally, the resulting dataset contains 28,709 images in training set, 3,589 images in validation set, and 3,589 images in test set. See Figure 3.2 for examples from the dataset with their corresponding emotion.

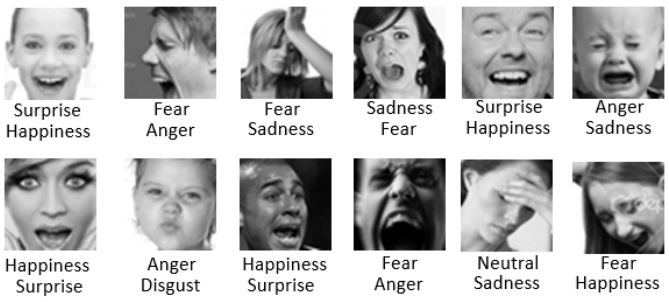

Figure 3.1: FER vs FER+ examples. Top labels are FER and bottom labels are FER+ (after majority voting). Figure taken from [3].

Figure 3.2: Examples of the Fer+ dataset. Each row consists of faces of the same emotion, starting from top row: neutral, happiness, surprise, sadness, anger, disgust, fear, and contempt.

3.2 Extended Cohn-Kanade (CK+)

The extended Cohn-Kanade (CK+) database consists of 593 sequences across 123 persons which are FACS coded at the peak frame. All sequences start from the neutral face and end to the peak expression. So for each sequence of images there is only 1 file with the Action Units. From those 593 sequences, 327 have emotion labels from 7 basic emotions (anger, contempt, disgust, fear, happy, sadness, surprise). See Figure 3.3 for examples from the dataset with their corresponding emotion and their action units. Also Table 3.1 depicts the action units along with their description and their connection with the basic emotional states that are used to our experiments.

Figure 3.3: Examples of the CK+ dataset. From left to right: (a) Anger AU 4, 7, 17, 23, 24, (b) Contempt AU 14, 15, 24, (c) Disgust AU 9, 15, 17, 24, (d) Fear AU 1, 4, 7, 20, 25, (e) Happy AU 6, 12, 16, 25, (f) Sadness AU 1, 2, 4, 15, 17, (g) Surprise AU 1, 2, 5, 25, 27

3.3 IMDB-WIKI 500+

Also we use the IMDB-WIKI dataset, the largest publicly available dataset of face images annotated with age and gender [33]. The dataset contains 460,723 face images from 20,284 celebrities from IMDB, and 62,328 from Wikipedia. A few examples are given in Figure 3.4.

Figure 3.4: Examples of the IMDB-WIKI dataset.

	NE	HA	SU	SA	AN	DI	FE	CO
FER2013	6198	8989	4002	6077	4953	547	5121	0
FER+	11660	9318	4377	4746	3244	303	1023	291
CK+	0	69	83	28	45	59	25	18

Table 3.2: Number of faces per emotion for each dataset. NE: Neutral, HA: Happiness, SU: Surprise, SA: Sadness, AN: Anger, DI: Disgust, FE: Fear, CO: Contempt

AU	Name	Associated emotions
1	Inner Brow Raiser	Surprise, Sadness, Disgust, Fear
2	Outer Brow Raiser	Surprise, Sadness, Fear
4	Brow Lowerer	Sadness, Anger, Disgust, Fear
5	Upper Lip Raiser	Surprise, Anger, Fear
6	Cheek Raiser	Happiness, Sadness
7	Lid Tightener	Fear
9	Nose Wrinkler	Disgust
10	Upper Lip Raiser	Disgust
11	Nasolabial Deepener	-
12	Lip Corner Puller	Happiness
13	Check Puller	-
14	Dimpler	Contempt
15	Lip Corner Depressor	Sadness, Anger, Disgust
16	Lower Lip Depressor	-
17	Chin Raiser	Sadness, Anger, Disgust
18	Lip Puckerer	-
20	Lip Stretcher	Fear
21	Neck Tightener	-
23	Lip Tightener	Anger
24	Lip Pressor	-
25	Lips Part	Happiness, Surprise
26	Jaw Drop	-
27	Mouth Stretch	-
28	Lip Suck	-
29	Jaw Thrust	-
31	Jaw Clencher	-
34	Cheek Puff	-
38	Nostril Dilator	-
39	Nostril Compressor	-
43	Eyes Closed	-

Table 3.1: Action unit codes with their description and their association with basic emotional states.

Chapter 4

Architectures

In 2012 AlexNet [23] won on ILSVRC2012 classification task by a large margin compared to the non deep learning algorithms. The goal of this classification challenge is to train a model that classifies an input into 1,000 categories. Models are trained on a very large dataset containing 1.2 million images, with 50,000 images for validation, and 100,000 for test set. The training set is derived from the ImageNet [7] project that contains images from almost 22,000 categories. The AlexNet contains 5 convolutional layers and 3 fully connected layers as depicted in Figure 4.1.

According to the universal approximation theory [18] a feed-forward network is able to approximate any measurable function to any desired degree of accuracy. However the layer can become very wide that is good at memorization prone to overfitting the data and not so good at generalization to new unseen data. Therefore, the trend after AlexNet was to increase the number of layers, so as to create deeper networks that are able to generalize better on unseen data. As a result of this tendency new architectures have been presented that achieved state-of-the-art performance on the contest and are going to be discussed to the following sections.

4.1 VGG

The VGG architecture was introduced by the VGG group at Oxford University in 2014 [36]. It simply replaces the large kernel filters from AlexNet from the first two convolutional layers with 3×3 kernel filters stacked on top of each other in increasing depth. The basic idea is that multiple smaller kernels are better than the one with larger kernel because multiple small sized kernels increase the depth of the network and as a result it can learn more complex features. Max pooling layers are used between the convolutional layers that reduce the number of network's parameters.

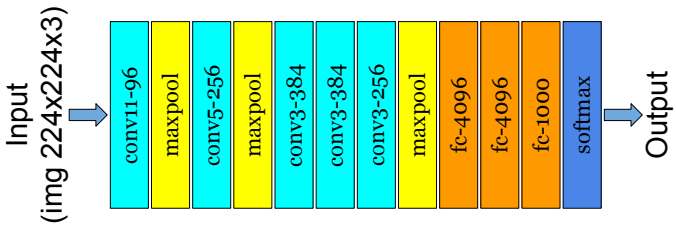

Figure 4.1: AlexNet architecture

Figure 4.2: VGG16 architecture

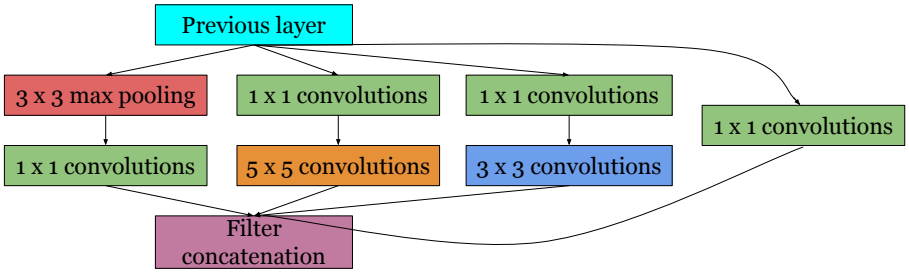

Figure 4.3: The inception module used in GoogLeNet.

The convolutional layers are followed by two fully connected layers each with 4,096 nodes, then by one fully connected layer with 1,000 nodes and finally by the softmax layer that assigns probabilities to the classes. So they introduced two versions of VGG network, one with 16 layers called VGG16 presented at Figure 4.2, and a deeper architecture with 19 layers. VGG network first introduced the idea that networks have to have a deep structure in order to extract complex features as you go deeper.

4.2 GoogLeNet/Inception

Besides the fact that VGG architecture achieved state-of-the-art results on ImageNet dataset, stacking all these layers and all these number of filters has a huge computational and memory cost. The GoogLeNet architecture [38] builds on the idea that most of the activations in a deep network are either zero or unnecessary because of correlations between them. GoogLeNet is a 22 layer CNN that firstly escaped from the idea that deep networks had to be built by simply stacking convolutional layers, pooling layers, and fully connected layers in a sequential structure. So they introduced the "inception" micro-architecture depicted in Figure 4.3. The "inception" module acts on the same input with different convolutional filters and does pooling at the same time. All results are then concatenated before being fed to the next layer of the network. This allows the model to extract features of different sizes.

4.3 Residual Networks

Residual networks were born from the simple observation that very deep networks perform worse from their shallower counterparts, that is, the performance with the addition of many layers saturates or even degrades rapidly. This observation is

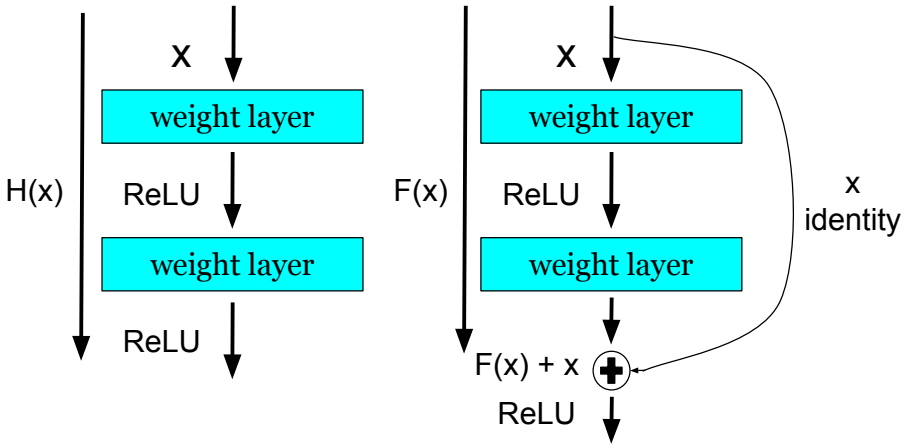

Figure 4.4: Normal CNN connections vs CNN with residual connections

counter-intuitive since we could stack layers with identity mappings on the network and the resulting architecture should perform the same. Each block in ResNet depicted in Figure 4.4 consists of a series of layers and a identity shortcut connection that adds the input of the block to its output. By doing this, ResNet architecture solved the problem of vanishing gradients with the simplest possible way. After that, this architecture became the hottest architecture in Computer Vision tasks and gained in popularity in the research community. Many architectures that are based on ResNet have been presented the following years and are presented below.

4.3.1 ResNet

He et al. [16] introduced the Residual Block depicted in Figure 4.5(a). However using this residual block, a very deep ResNet arcchitecture gave worse results compared to the shallower 110 layer ResNet. So they refined the residual block and proposed a new version depicted on Figure 4.5(b). With experiments they trained a very deep 1001 layer ResNet that achieved better results by its shallower counterparts.

4.3.2 DenseNet

Huang et al. [19] proposed a novel architecture called DenseNet that exploits the effects of shortcut connections. In this architecture all layers are connected to each other. Every layer receives as input the feature maps from all previous layers and sends its output to all subsequent layers. Figure 4.6 gives an example of the network. The authors claim that this architecture except from solving the vanishing gradient problem, it encourages feature re-use, and reduces the number of parameters. However, DenseNets can require a significant amount of GPU memory despite the fact that they have less parameters. DenseNets have much more connections than ResNets, and since Back-propagation requires all layers' output to be in memory, that is the main reason for memory bottlenecks.

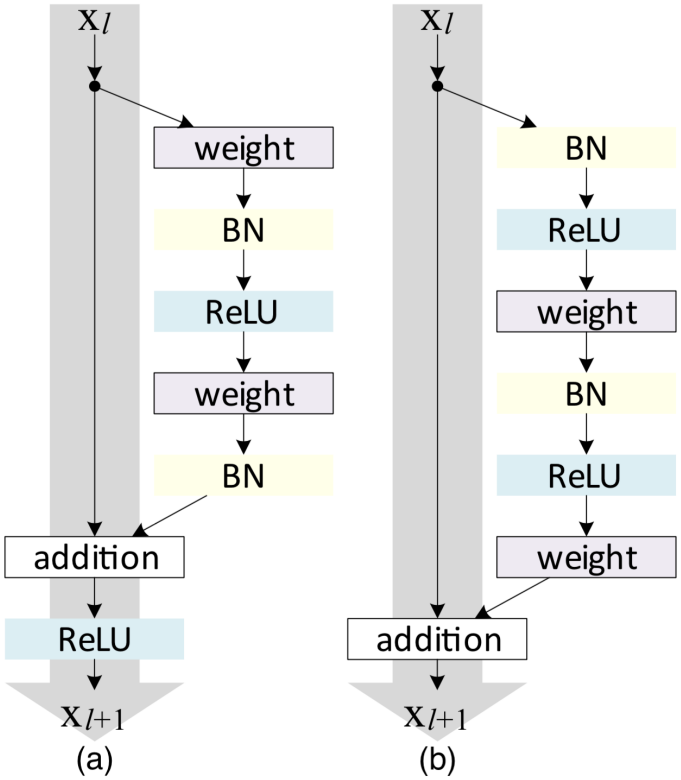

Figure 4.5: (a) original Residual Unit, (b) enhanced Residual Unit. Extracted from [17]

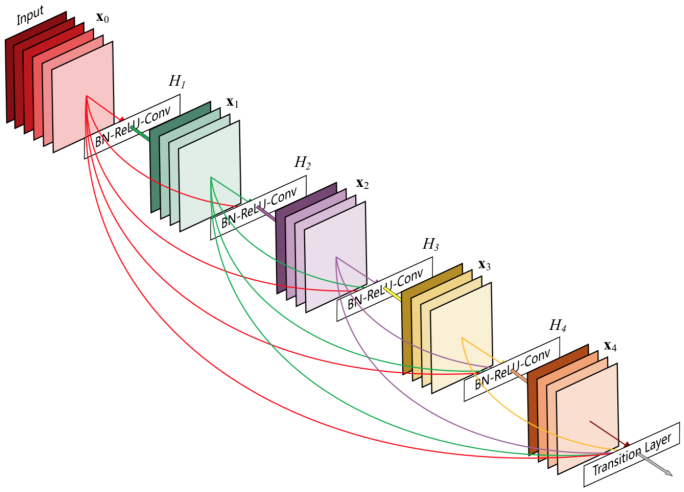

Figure 4.6: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes as input the feature maps from all previous layers. Extracted from [19]

4.3.3 Wide ResNet

Zagoruyko and Komodakis [46] to tackle the computational and memory contraints of the DenseNet network, instead of increasing the depth (number of layers) of the network, they experimented by increasing the width (number of neurons per layer or for convolutional layers the number of feature maps per layer) and also using dropout to regularize the wider networks. So a Wide ResNet is just a ResNet with more feature maps in its convolutional layers. They experimented with architectures of different depth and width with the same number of parameters and showed that the wider version performs better on CIFAR-10, CIFAR-100 [22], and SVHN [28]. As mentioned in their paper, thin and deep residual networks with small kernels are against the nature of GPU computations because of the sequential structure, as opposed to wide networks that are many times more efficient than the deep and thin network that achieves the same performance.

Chapter 5

Transfer Learning

Transfer learning is a machine learning technique in which the knowledge gained from one domain is transferred to a different but related problem. Recently, transfer learning has gained in popularity in computer vision problems after the success of [11], [32], and [34]. Yosinski et al. [44] defines a way to assess the transferability of features from different layers in deep neural networks and shows two cases where the performance drops after feature transferring, (1) when transferring features that are highly specialized in the first task, and (2) from optimization difficulties due to the splitting of the original network that leads to fragile co-adaption between units in consecutive layers. Azizpour et. al [1] identified several factors that affects the transferability of features. Also they give advice on how to maximize the performance of a generic CNN when applied to a new task based on the similarity between the two tasks.

As discussed earlier in the case of image classification, CNNs extract features later used by the fully connected layers to classify the image (Figure 5.1). Multiple CNNs (VGG16, VGG19, ResNet50, InceptionV3, etc.) are available that are trained on large datasets such as ImageNet. These architectures have been trained on millions of images and classify the images to one of 1000 categories, such as car, airplane, etc. So transfer learning can be used in two ways:

- The output of a layer can be used as features that are fed as input to a separate classifier (Figure 5.2).
- Fine-tuning the entire network if the new dataset is similar to the original dataset, or freezing the first few layers as these layers detect edges and blobs and fine-tune the rest of the network (Figure 5.3).

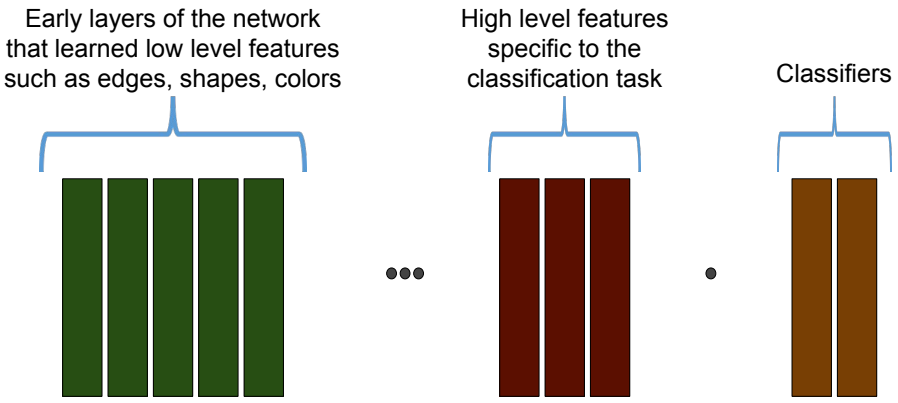

Figure 5.1: Feature extraction from Convolutional Neural Network.

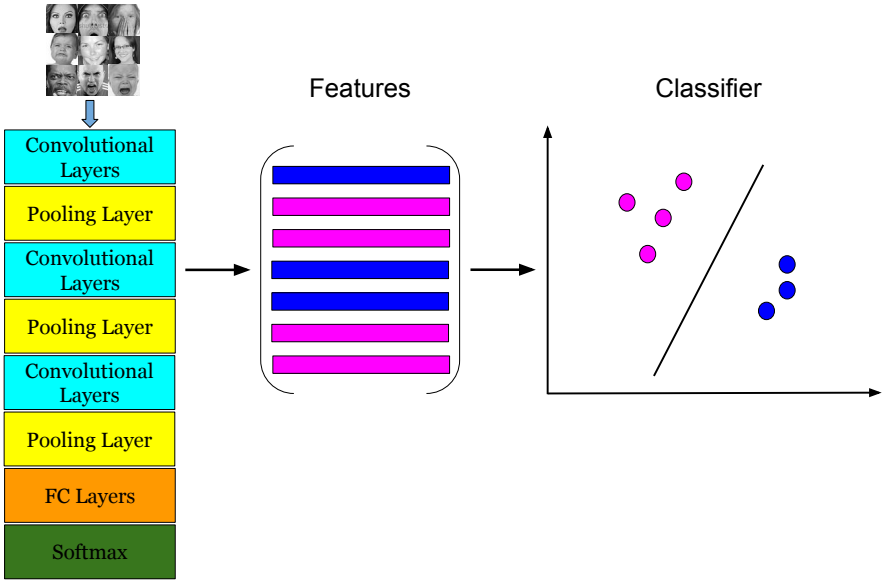

Figure 5.2: Feature extraction from a pre-trained Convolutional Neural Network.

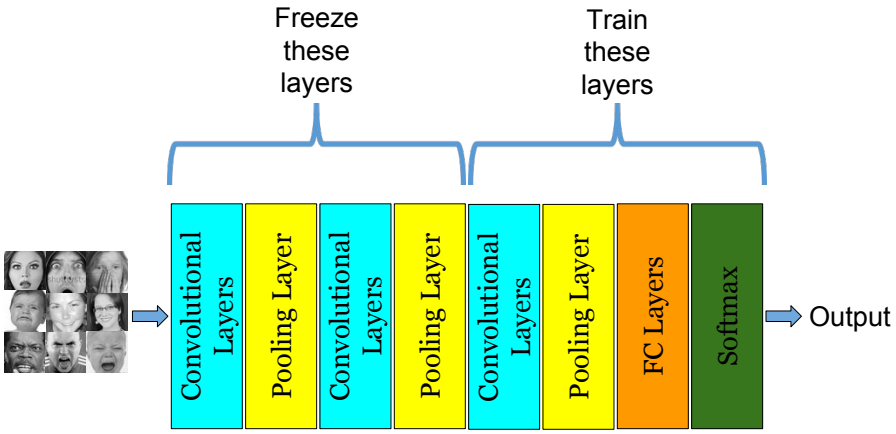

Figure 5.3: Fine-tuning the pre-trained network.

Chapter 6

Multi-task learning

Multi-task learning has recently gained in popularity in machine learning after the success in many different domains such as speech recognition [8], computer vision [15], natural language processing [6], and drug discovery [31]. It aims at leveraging information from relevant tasks in order to improve the generalisation performance in all tasks. Multi-task learning tries to mimic the way people learn, that is, by transferring knowledge from one task to another when the tasks are related. So it is useful to learn multiple tasks simultaneously since the knowledge from one task can be utilised in the other task.

Multi-task learning is related to other areas of machine learning including transfer-learning described on chapter 5. The main difference between those two techniques is that in transfer-learning, the transfer of knowledge between related tasks is done sequentially, and the main goal is to improve the performance of the target task, while in multi-task learning the learning process is parallel for all tasks and the goal is to improve the performance for all tasks. Generally the goal of multi-task learning is to improve the generalisation performance for all related tasks. There are two ways to perform multi-task learning in deep networks. In the context of Deep learning, multi-task learning is done with either hard or soft parameter sharing of the layers.

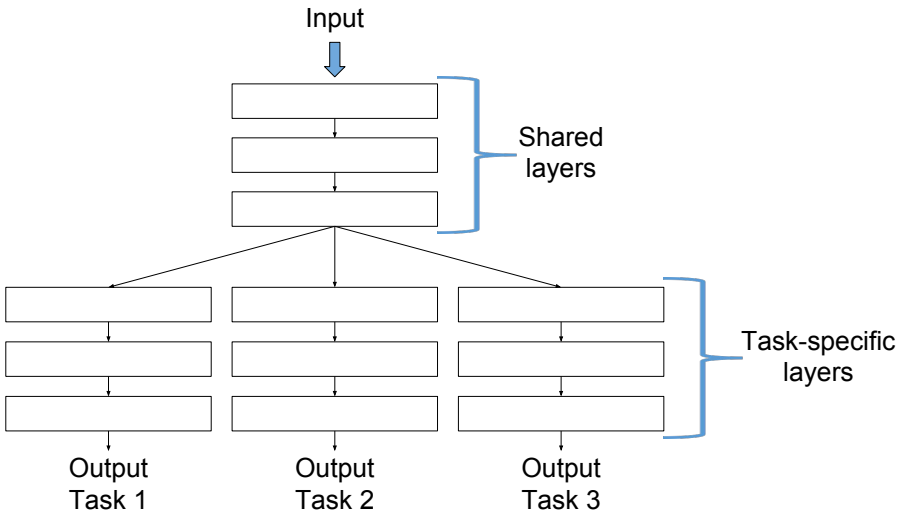

Figure 6.1: Hard parameter sharing for multi-task learning in deep neural networks

6.1 Hard parameter sharing

Hard parameter sharing [5] is applied by sharing some layers between all tasks and keeping some task-specific layers for each task as depicted in Figure 6.1.It is by far the most commonly used approach to multi-task learning. This approach reduces the risk of overfitting because it tries to find a common representation that works well for all tasks.

6.2 Soft parameter sharing

In soft parameter sharing each task has its own set of parameters and its own model. The distance between the parameters of each task is then regularized for example by using $\ell 2$ norm as [12] in order to force the parameters to be as similar as possible.

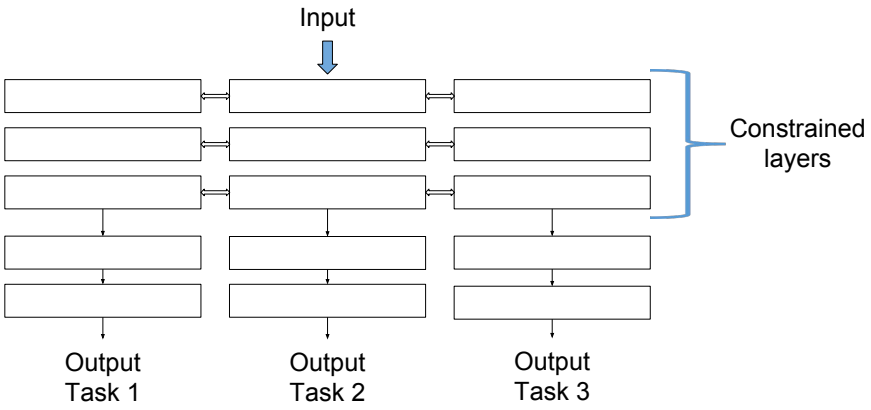

Figure 6.2: Soft parameter sharing for multi-task learning in deep neural networks

Chapter 7

Experiments & Results

7.1 State-of-the-art Architecture

The custom version of VGG network depicted in Figure 7.1 achieves the state-of-the-art accuracy of 84.9% on the FER+ dataset. This network has 10 convolutional layers, interleaved with max pooling and dropout layers. More concretely, there are 2 convolutional layers after the input with 64 filters of size 3×3. After those 2 layers, max pooling and dropout layers with dropout rate of 25% follow. The structure repeats but changes in the number of feature maps per convolutional layer and the number of convolutional layers. After the 10 convolutional layers 2 dense layers are added each with 1024 nodes and each followed by a dropout rate of 50%. The final dense layer has 8 nodes equal to the number of classes and is followed by a softmax layer that gives the probability for every emotion category. This architecture is trained from scratch, and during training they augment the data on the fly by applying simple affine transformations. Since the dataset is annotated by 10 taggers, they could generate a probability distribution for every image where the probability for a specific emotion is equal to the number of annotators that voted for this emotion divided by the total number of annotators which is 10. So for the n^{th} image in the dataset the crowdsourced emotion distribution is p_k^n with $k = 1...8$, where:

$$\sum_{k=1}^{8} p_k^n = 1$$

The loss function used for training this architecture is the categorical cross entropy given by:

$$L(t) = -\sum_{n=1}^{N}\sum_{k=1}^{8} p_k^n \log q_k^n$$

where p_k^n is the probability given by the annotators for the image n and for the emotion k and q_k^n is the probability given by the softmax layer for the same image and emotion.

The metric that they used to evaluate their model is the accuracy which is the percentage of correctly classified instances and is given by the following formula:

$$Accuracy = \frac{TP+TN}{N}$$
where TP is the true positives, TN is the true negatives
and N the number of images in the dataset

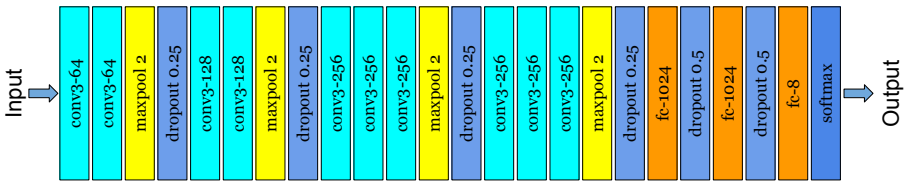

Figure 7.1: State of the Art Architecture

In this project, we experiment with the latest deep convolutional neural networks as discussed in chapter 4 by training them from scratch with the FER+ dataset. Also we experiment with the two most popular techniques in Computer Vision, namely Transfer learning and Multi-task learning that are beneficial in many cases discussed in chapters 5 and 6 respectively. Furthermore, we use Action Units discussed in section 1 as additional information which encode facial muscle movement and can help to improve the performance. Also as [3] did, we apply data augmentation to our dataset by applying geometric transformations to the images. Data augmentation is a methodology that generates new data from our existing dataset. By using this technique the model is able to generalise well during the test phase, and also is very useful to combat imbalanced datasets which is the case for the FER+ dataset. More concretely we use the following transformations for all the experiments:

- Scaling
- Rotation
- Translation
- Horizontal flipping

Also in every architecture we experiment with different values for the following hyperparameters:

- Optimizer (Adam [21], Stochastic gradient descent)
- Learning rate (0.1, 0.01, 0.001, 0.0001)
- Batch size (32, 64, 128)
- Dropout rate (0.25, 0.3, 0.5)
- Decay (1e-4, 1e-5, 1e-6)

Finally during the training phase we use as the loss function the *categorical cross entropy* and the *accuracy* as the metric in order to have a direct comparison with the state of the art accuracy.

7.2 VGG-13

VGG13 (Figure 7.2) is the shallowest version of the VGG architecture with 10 convolutional layers. The word *conv3-512* from the figure's box means that the convolutional layer uses 512 filters of size 3×3. The word *maxpool2* means that the size of max pooling layer is of size 2×2, and the word *fc-4096* means that the layer is fully connected with 4096 nodes. The hyperparameters that achieve the best test accuracy are:

- Optimizer: Stochastic gradient descent
- Learning rate: 0.01
- Batch size: 128
- Dropout rate: Not available in this architecture
- Decay: 1e-6

The result can be found in table 7.1.

7.3 VGG-16

We try a deeper version of VGG architecture with 3 additional convolutional layers with 512 feature maps each depicted on Figure 7.2.

- Optimizer: Stochastic gradient descent
- Learning rate: 0.01
- Batch size: 128
- Dropout rate: Not available in this architecture
- Decay: 1e-6

The result can be found in table 7.1.

7.4 VGG-19

We try the deepest version of the VGG architecture with 16 convolutional layers and 3 fully connected layers depicted on Figure 7.2.

- Optimizer: Stochastic gradient descent
- Learning rate: 0.01
- Batch size: 128
- Dropout rate: Not available in this architecture
- Decay: 1e-6

The result can be found in table 7.1.

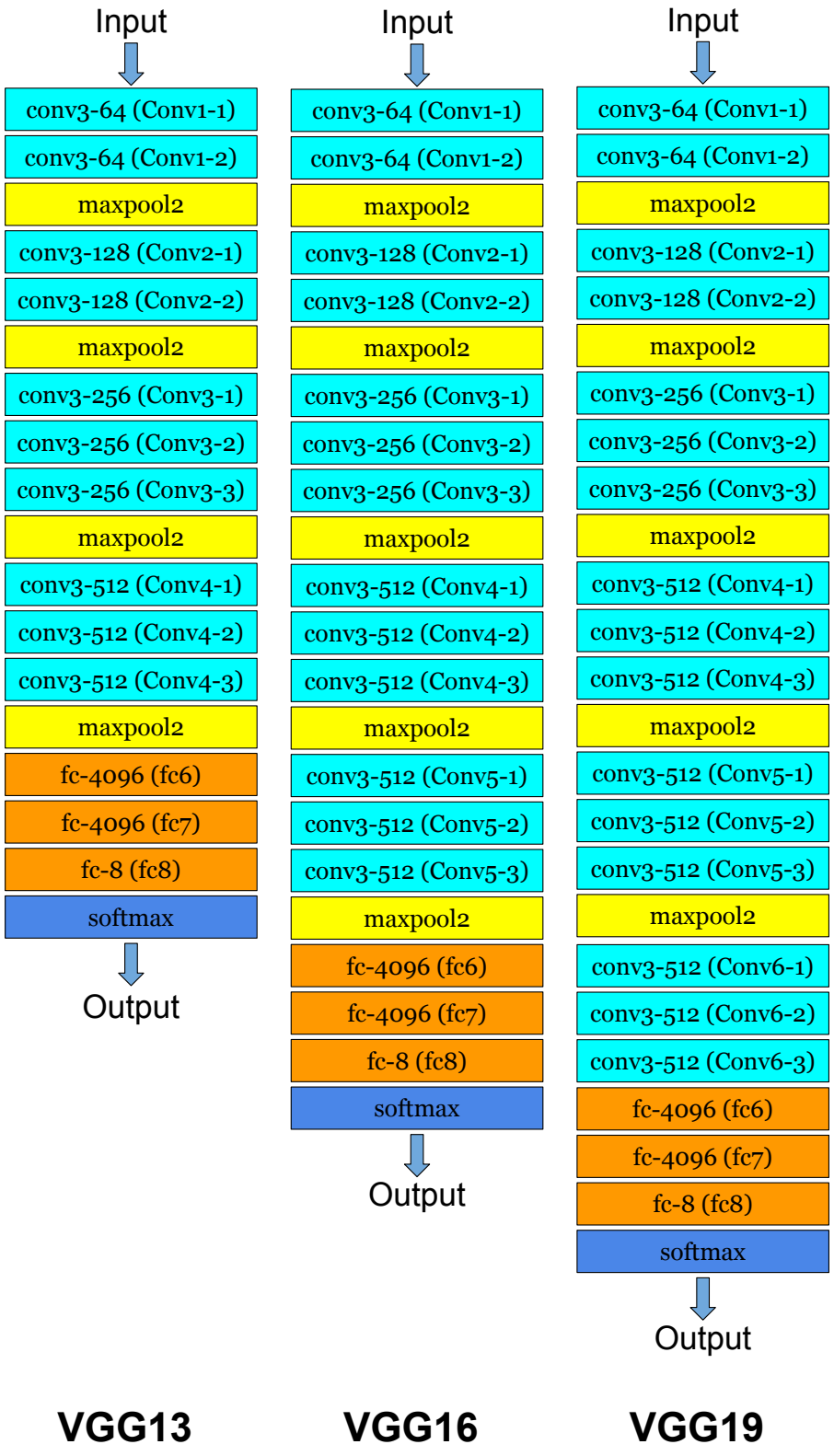

Figure 7.2: VGG Architectures

7.5 ResNet-18

We evaluate a 18-layer residual network. See Figure 7.3 for details concerning the architecture.

- Optimizer: Adam
- Learning rate: 0.001
- Batch size: 128
- Dropout rate: Not available in this architecture
- Decay: 0.0

The result can be found in table 7.1.

7.6 ResNet-34

We evaluate a 34-layer residual network. See Figure 7.3 for details concerning the architecture.

- Optimizer: Adam
- Learning rate: 0.001
- Batch size: 128
- Dropout rate: Not available in this architecture
- Decay: 0.0

The result can be found in table 7.1.

7.7 ResNet-50

We evaluate a 50-layer residual network. See Figure 7.3 for details concerning the architecture.

- Optimizer: Stochastic gradient descent
- Learning rate: 0.001
- Batch size: 128
- Dropout rate: Not available in this architecture
- Decay: 0.0

The result can be found in table 7.1.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 7.3: Examples of conversion from grayscale to RGB images

7.8 DenseNet

Unfortunately we couldn't experiment with this architecture, since it is very computationally expensive and as a result we couldn't train with a batch size larger than 8.

7.9 Wide ResNet

As discussed in chapter 4 Wide ResNet is a novel architecture that decreases the depth of the network while increasing the width of the convolutional layers. By doing this, we can achieve superior performance over the commonly used very deep and thin networks. After many experiments, we end up with an architecture that is 28 layers deep and is 4 times wider than the common ResNet network. By using this network with the following hyper-parameters we achieved state-of-the-art accuracy to our main dataset.

- Optimizer: Stochastic gradient descent
- Learning rate: 0.1
- Batch size: 128
- Dropout rate: 0.3
- Decay: 0.0

The result can be found in table 7.1.

7.10 Transfer Learning

As discussed in chapter 5 there are 2 ways to perform transfer learning, either you use a pre-trained network to extract features that you feed to a classifier to do the classification, or you use the entire pre-trained architecture, freeze some layers and train the rest of the layers. In our experiments we use architectures that are trained on ImageNet and for face recognition task namely VGGFace [29].

Figure 7.4: Examples of conversion from grayscale to RGB images

7.10.1 Using pre-trained CNN features

Despite the fact that CNN are complex models and many details on how these models work remain a mystery, we know that the convolutional layers closer to the input represent low-level image features such as edges, while higher level convolutional layers represent more complex features such as faces, emotions, etc. The final fully connected layers are generally assumed to contain information that is relevant to the task. So the idea of this method as discussed in chapter 5 is to use the CNN as a feature generator that extracts a set of features for every input. Those features are fed to a Linear SVM classifier that does the classification.

VGG16 pretrained on ImageNet

We use the pre-trained on ImageNet VGG16 network depicted on Figure 7.2 as the feature generator. Since the original network was trained with images of dimensions $224 \times 224 \times 3$, we have to convert our images from grayscale to rgb and resize them from 48×48 pixels to 224×224 pixels. Examples of this conversion are depicted on Figure 7.4. From the *fc6* layer we extract 4096 features that we use to train an SVM classifier. Furthermore we extract features from the *fc7* layer and again we train an SVM classifier. The algorithm is depicted in Figure 7.5. The results can be found in Table 7.2.

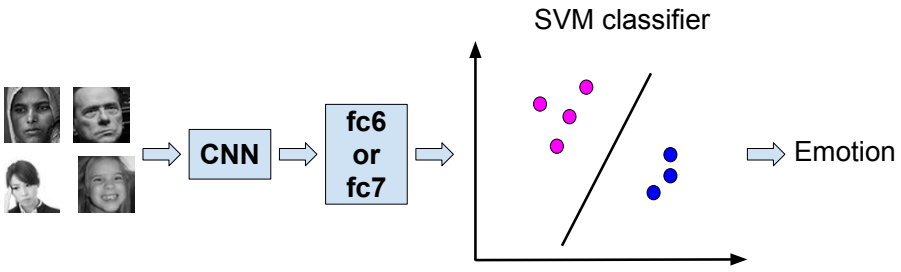

Figure 7.5: Illustration of the proposed method for transfer learning.

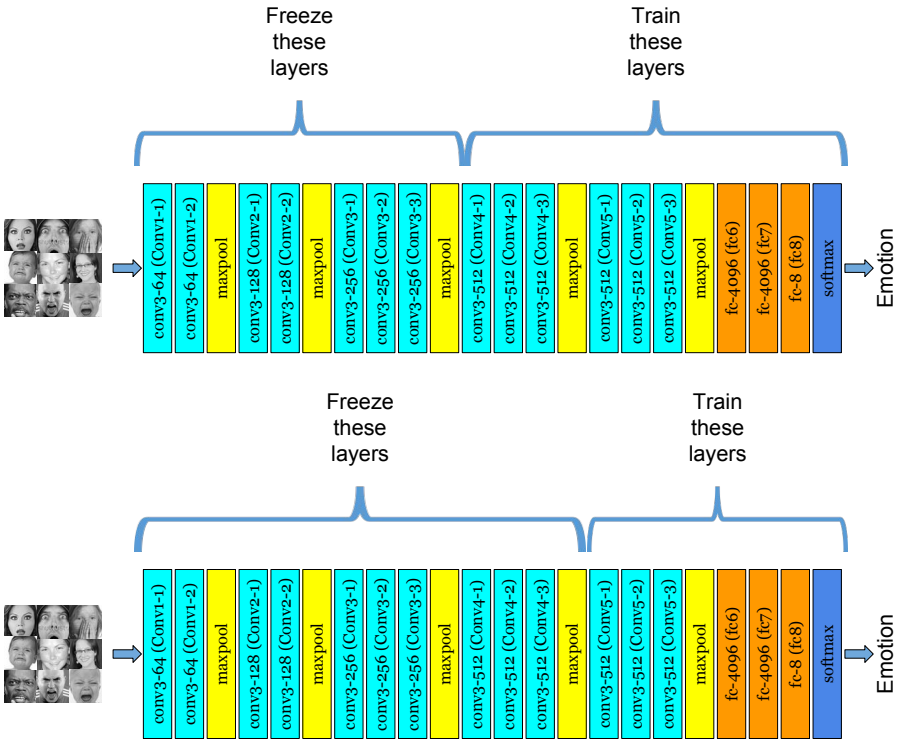

Figure 7.6: Fine-tuning VGG16 architecture.

ResNet50 and InceptionV3 pretrained on ImageNet

As before we use these pre-trained networks to extract features from the last layer and follow the same methodology as before. The results can be found in Table 7.2.

VGG16 pretrained for face recognition

We use the pre-trained for face recognition VGG16 network and follow the same methodology as with the VGG16 model pre-trained on ImageNet. Again we extract features from the *fc6* and *fc7* layers and train an SVM classifier. The results can be found in Table 7.2.

7.10.2 Fine-tuning

VGG16 pretrained on ImageNet and for Face Recognition

In this workflow we use the pre-trained on ImageNet and for Face Recognition VGG16 networks. The idea is to not train from scratch the entire network but to fine-tune some of the higher-level layers during backpropagation, while keeping the rest of the layers intact. As discussed in chapter 5 this is motivated by the fact that the earlier layers of a convolutional neural network learn some generic features (edges, blobs, colors, etc.) that are useful to many tasks. So we experiment by freezing two different sets of convolutional layers as depicted in Figure 7.6.

CNN	
Architecture	**Accuracy**
VGG13	84.45
VGG16	84.43
VGG19	84.2
ResNet18	82.57
ResNet34	82.82
ResNet50	82.37
GoogLeNet	35.64
DenseNet	-
Wide ResNet	**86.3**

Table 7.1: Accuracies from Convolutional Neural Networks

7.11 Multi-task Learning

In transfer learning we care about getting good results on our target domain. On the contrary in multi-task learning the goal is to exploit the synergy effect between closely related tasks and improve the accuracy on all tasks. The starting point for doing multi-task learning is the architecture that achieved state-of-the-art result on the FER+ dataset depicted on Figure 7.1. After many experiments between we conclude to the architecture depicted on Figure 7.7. The network has 3 branches as the number of tasks. We train this network with 2 different datasets. The branch that does the emotion recognition is trained with the FER+ dataset, and the other 2 branches with the IMDB-WIKI 500+ described on Chapter 3. The result of this method can be found in Table 7.3.

7.12 Discussion on the Results

As we can see from the tables 7.1, 7.2, and 7.3, training deep neural networks from scratch achieves much higher performance than transfer learning and multi-task learning. Also we observe that as we increase the depth of VGG and ResNet architectures the performance degrades. This is also obvious by the performance of GoogLeNet architecture which is a very deep network. Maybe this is caused by the fact that our dataset is small for these deep architectures, and that's why a wider and shallower network like Wide ResNet can achieve such a good performance. Transfer learning doesn't work in our case. Maybe this is caused by the fact that our dataset consists of low resolution 48×48 pixels images that need to be resized to 224 pixels and converted to images with 3 channels. As depicted in Figure 7.4 the conversion is far from the ground truth.

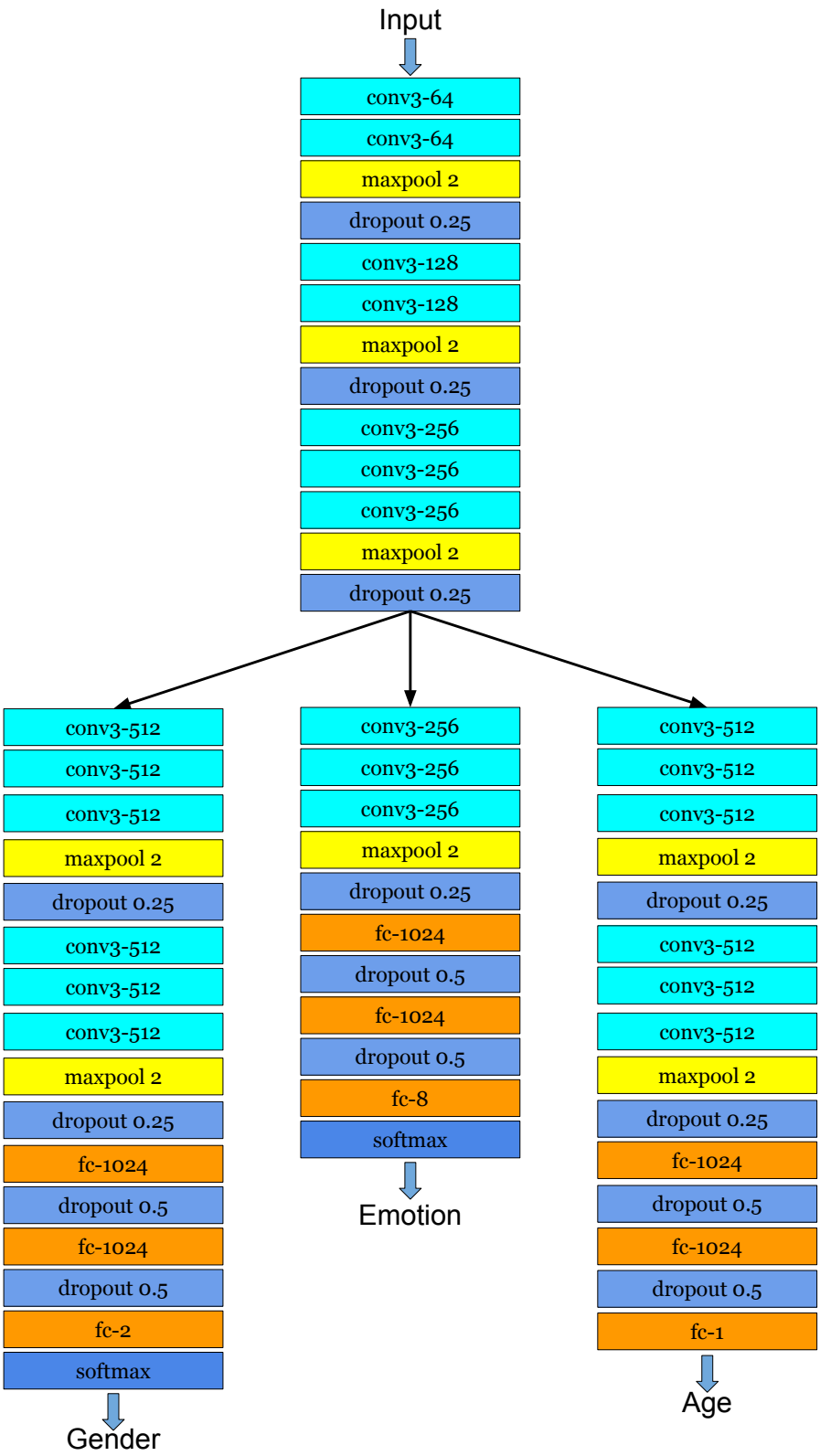

Figure 7.7: Multi-task learning

Transfer Learning	
Architecture	**Accuracy**
Fine-tuning VGG16, ImageNet, conv3_3	46.39
Fine-tuning VGG16, ImageNet, conv4_3	42.21
Fine-tuning VGG16, Face Recognition, conv3_3	**53.52**
Fine-tuning VGG16, Face Recognition, conv4_3	45.67
fc6 Convolutional features VGG16, ImageNet	34.22
fc7 Convolutional features VGG16, ImageNet	29.25
fc6 Convolutional features VGG16, Face Recognition	27.2
fc7 Convolutional features VGG16, Face Recognition	23.1
ResNet50, ImageNet	21.13
InceptionV3, ImageNet	19.14

Table 7.2: Accuracies from Transfer Learning

Multi-task Learning	
Architecture	**Accuracy**
3 Tasks (Emotion, Gender, Age)	85.2

Table 7.3: Accuracy from Multi-task learning

Chapter 8

Best Architecture

Wide ResNet achieves an accuracy of 86.3% which is much higher than the state-of-the-art accuracy on the FER+ dataset. Also we observe that the top-2 accuracy is 95.69%, and the top-3 accuracy is 98.35% which are both very high. So the idea is to create an additional model that will give an extra "opinion" about the predicted emotion when our main network (Wide ResNet) has no clear winner (when the probabilities of the top labels are close to each other). To create a new model that will provide an extra assistance to our main model we use action units which encode facial muscle movement. More concretely, we use OpenFace [2], an open source tool capable of facial landmark detection, head pose estimation, facial action unit recognition, and eye-gaze estimation. By using this tool we can create a separate dataset with action units to serve as the features and the emotions to be the labels. The next step is to train an SVM classifier for every pair of emotions. We only take as features, the action units that are associated with the two emotions of the binary classification as depicted in table 3.1. Also we train an SVM classifier for every 3 emotions. By using these two extra models we increase the accuracy from 86.3% to 86.6%. The algorithm is depicted in Figure 8.1.

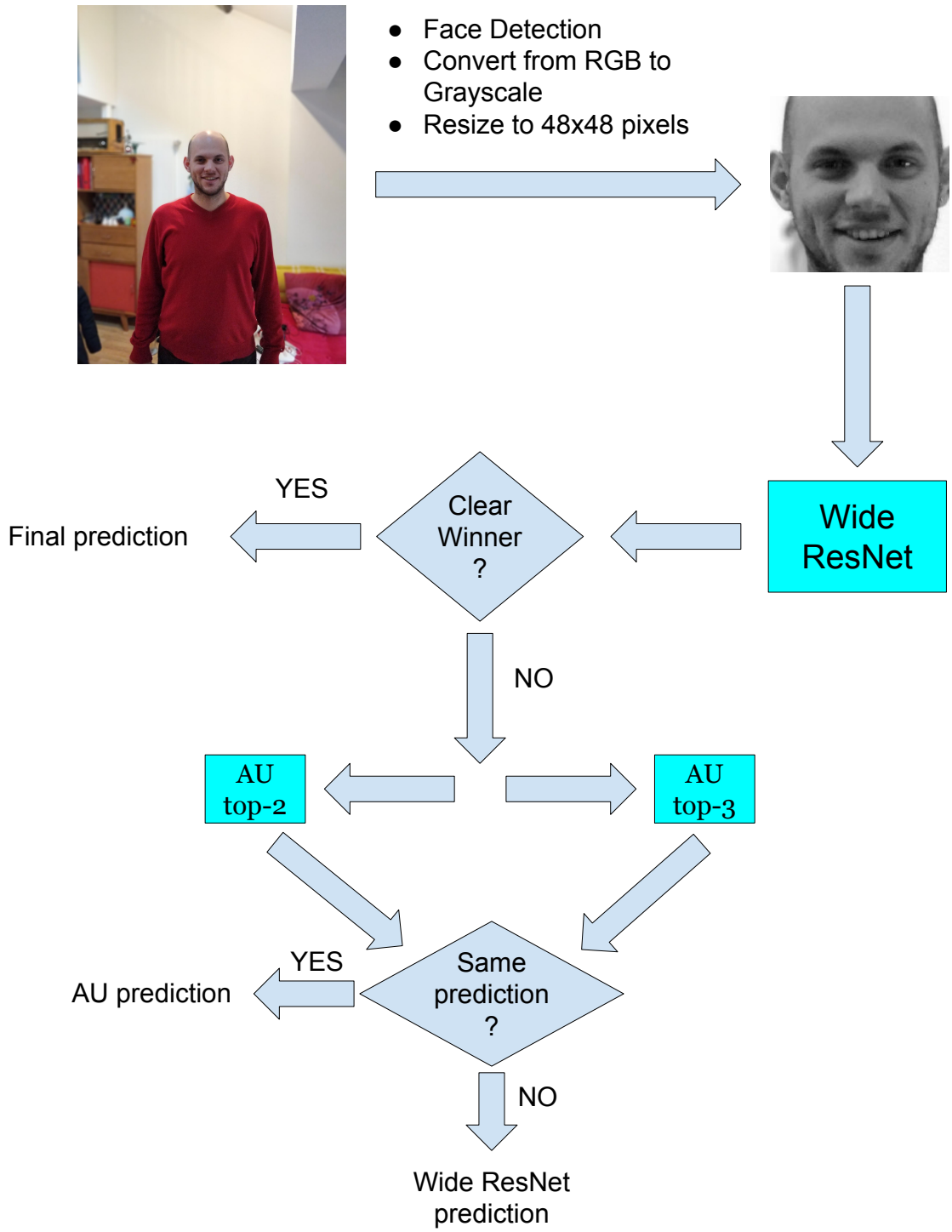

Figure 8.1: Final Algorithm.

Chapter 9

Conclusion

In this project, we compare different Convolutional Neural Networks on the FER+ dataset. In every architecture we experiment with many different sets of hyper-parameters until we get the best accuracy on the validation set. We show that for the case of a small dataset usually a shallower and wider network perform better than its deeper counterparts. Moreover, the fact that our dataset consists of grayscale images has a negative impact on the performance of transfer learning. Finally we propose a novel algorithm that provides state-of-the-art results on emotion recognition. The key idea is that, two models based on actions units can help the deep neural network to generalise better on unseen data.

Bibliography

[1] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson. Factors of transferability for a generic convnet representation. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1790–1802, 2016.

[2] T. Baltrušaitis, P. Robinson, and L.-P. Morency. Openface: an open source facial behavior analysis toolkit. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–10. IEEE, 2016.

[3] E. Barsoum, C. Zhang, C. C. Ferrer, and Z. Zhang. Training deep networks for facial expression recognition with crowd-sourced label distribution. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, pages 279–283. ACM, 2016.

[4] S. Berretti, A. Del Bimbo, P. Pala, B. B. Amor, and M. Daoudi. A set of selected sift features for 3d facial expression recognition. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 4125–4128. IEEE, 2010.

[5] R. Caruana. Multitask learning: A knowledge-based source of inductive bias1. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48. Citeseer.

[6] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[8] L. Deng, G. Hinton, and B. Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8599–8603. IEEE, 2013.

[9] O. Déniz, G. Bueno, J. Salido, and F. De la Torre. Face recognition using histograms of oriented gradients. *Pattern Recognition Letters*, 32(12):1598–1603, 2011.

[10] A. Dhall, R. Goecke, J. Joshi, K. Sikka, and T. Gedeon. Emotion recognition in the wild challenge 2014: Baseline, data and protocol. In *Proceedings of the 16th International Conference on Multimodal Interaction*, pages 461–466. ACM, 2014.

[11] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[12] L. Duong, T. Cohn, S. Bird, and P. Cook. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 845–850, 2015.

[13] P. Ekman and W. V. Friesen. Constants across cultures in the face and emotion. *Journal of personality and social psychology*, 17(2):124, 1971.

[14] P. Ekman and W. V. Friesen. *Facial Action Coding System: Investigatoris Guide*. Consulting Psychologists Press, 1978.

[15] R. Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[18] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[19] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.

[20] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015.

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[24] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[25] P. Lucey, J. Cohn, S. Lucey, I. Matthews, S. Sridharan, and K. M. Prkachin. Automatically detecting pain using facial actions. In *Affective Computing and Intelligent Interaction and Workshops, 2009. ACII 2009. 3rd International Conference on*, pages 1–8. IEEE, 2009.

[26] A. Mollahosseini, D. Chan, and M. H. Mahoor. Going deeper in facial expression recognition using deep neural networks. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–10. IEEE, 2016.

[27] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[28] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[29] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015.

[30] O. M. Parkhi, A. Vedaldi, A. Zisserman, et al. Deep face recognition. In *BMVC*, volume 1, page 6, 2015.

[31] B. Ramsundar, S. Kearnes, P. Riley, D. Webster, D. Konerding, and V. Pande. Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072*, 2015.

[32] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 512–519. IEEE, 2014.

[33] R. Rothe, R. Timofte, and L. Van Gool. Dex: Deep expectation of apparent age from a single image. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 10–15, 2015.

[34] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[35] C. Shan, S. Gong, and P. W. McOwan. Facial expression recognition based on local binary patterns: A comprehensive study. *Image and Vision Computing*, 27(6):803–816, 2009.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.

[39] Y. Tang. Challenges in representation learning: Facial expression recognition challenge implementation. *University of Toronto*, 2013.

[40] Y.-I. Tian, T. Kanade, and J. F. Cohn. Recognizing action units for facial expression analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 23(2):97–115, 2001.

[41] A. Vinciarelli, M. Pantic, and H. Bourlard. Social signal processing: Survey of an emerging domain. *Image and vision computing*, 27(12):1743–1759, 2009.

[42] E. Vural, M. Çetin, A. Erçil, G. Littlewort, M. Bartlett, and J. Movellan. Automated drowsiness detection for improved driving safety. 2008.

[43] Z. Wang and Z. Ying. Facial expression recognition based on local phase quantization and sparse representation. In *Natural Computation (ICNC), 2012 Eighth International Conference on*, pages 222–225. IEEE, 2012.

[44] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[45] Z. Yu and C. Zhang. Image based static facial expression recognition with multiple deep network learning. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pages 435–442. ACM, 2015.

[46] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[47] M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*, 2013.

[48] Z. Zhang. Feature-based facial expression recognition: Sensitivity analysis and experiments with a multilayer perceptron. *International journal of pattern recognition and Artificial Intelligence*, 13(06):893–911, 1999.

[49] G. Zhao and M. Pietikainen. Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):915–928, 2007.