Honeywell

DHBW
Duale Hochschule
Baden-Württemberg
Mannheim

Duale Hochschule Baden-Württemberg
Mannheim

# Bachelor Thesis

**Neural Networks for Tool Image Classification**

## Course of Study: Business Informatics

Field of Study: Software Engineering


| | |
|---|---|
| Author: | Fabian Wolf |
| Matriculation Number: | 7345461 |
| Company: | Movilizer GmbH |
| Department: | Research and Development |
| Course of Study: | WWI17SEC |
| Head of Course of Study: | Prof. Dr.-Ing. habil. Dennis Pfisterer |
| Scientific Supervisor: | M.Sc. Boas Bamberger<br>bamberger@uni-mannheim.de<br>+49 621 181 1562 |
| Business Supervisor: | Oliver Erlenkaemper<br>oliver.erlenkaemper@honeywell.com<br>+49 621 150 207 36 |
| Processing Period: | 02 March 2020–11 May 2020 |

# Abstract


| | |
|---|---|
| Title | Neural Networks for Tool Image Classification |
| Author: | Fabian Wolf |
| Course of Study: | WWI17SEC |
| Company: | Movilizer GmbH |



The augmented reality market is estimated to reach 83.5 billion USD by 2021.[1] According to Boston Consulting Group, Accenture, Mc Kinsey, and others, augmented reality solutions for field workers are an important field within that market.[2] An augmented reality solution for field workers requires software perceiving the environment of field workers. A sub-task of that perception is to classify tools of different classes. This paper determines the best-performing neural network for tool image classification. The best-performing neural network for tool image classification is determined in the course of an experiment. The experiment trains and evaluates state-of-the-art neural networks for image classification on a dataset constructed by this paper. The state-of-the-art neural networks for image classification are determined in the course of a literature review conducted by this paper. This paper found that, in general, not only one neural network is suitable for tool image classification, but several neural networks are suitable for tool image classification. Especially, ResNet-152, ResNeXt-101, and DenseNet-264 were proven to be suitable for tool image classification. Furthermore, this paper introduces a novel dataset for tool image classification called Tool Image Classification Dataset (TIC Dataset). This paper hopes to foster further research in the field of computer vision by providing the TIC Dataset publicly available under the Creative Commons Attribution Share Alike 4.0 International License.


[1] Statista 2019.

[2] Ernst & Young 2019a; Ernst & Young 2019b; Detzel et al. 2018; Shook and Knickrehm 2017; Guy et al. 2019.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Acronyms

| | |
|---|---|
| **CNN** | Convolutional Neural Network |
| **ReLU** | Rectified Linear Unit |
| **RGB** | Red, Green, and Blue |
| **RMSProp** | Root Mean Square Propagation |
| **SGD** | Stochastic Gradient Descent |
| **TIC Dataset** | Tool Image Classification Dataset |

# 1 Introduction

## 1.1 Motivation

> It might take two hours for an experienced field technician to fix a broken MRI. But a pair of smart glasses that displays step-by-step instructions in the techs field of vision could help shave as much as 50% from the time it takes to diagnose the problems and make needed repairs. (Detzel et al. 2018)

According to Boston Consulting Group, Accenture, Mc Kinsey, and others, augmented reality solutions for field workers are an emerging market.[3] The augmented reality market is estimated to reach 83.5 billion USD by 2021.[4] Movilizer GmbH approaches this trend with its connected solutions.[5] An augmented reality solution for field workers requires software perceiving the environment of field workers. A sub-task of that perception is to distinguish tools of different classes. For example, a software displaying step-by-step instructions in the field of vision of a field worker needs to distinguish a screwdriver from a wrench when telling the field worker to tighten the screw with the wrench lying on the ground next to him instead of the screwdriver in his hand. Distinguishing between tools of different classes is called tool image classification. Image classification in general is mostly solved by neural networks.[6] This paper determines the best-performing neural network for tool image classification. Furthermore, this paper introduces a novel tool image classification dataset called TIC Dataset.

## 1.2 Problem Statement

Tool image classification is the problem of assigning the correct class to an image of a tool, see Section 2.5. An image of a tool is a close-up image of exactly one tool from an arbitrary angle with an arbitrary background. To classify an image, a neural network receives an image and returns the class probabilities. The class with the

[3] Ernst & Young 2019a; Ernst & Young 2019b; Detzel et al. 2018; Shook and Knickrehm 2017; Guy et al. 2019.
[4] Statista 2019.
[5] Bonner 2018; Honeywell International Inc. 2018.
[6] El-Amir and Hamdy 2020; LeCun, Bengio, and G. Hinton 2015a; P. Singh and Manure 2020; Michelucci 2019; Gad 2018; Kapur 2017.

highest probability is the prediction of the neural network for that image. A $w$-by-$h$-pixel image is formatted as three-dimensional array. The array contains one subarray for each pixel. The $X$ and $Y$ coordinates of a pixel in an image correspond to the indices of the subarray containing the $d$ color values of that pixel. Consequently, the three-dimensional array is of shape $w \times h \times d$.[7] For example, a three-color, 224-by-224-pixel image is formatted as an array of shape $224 \times 224 \times 3$. The class probabilities are formatted as a one-dimensional array. Each index of that array corresponds to one class. The element at a given index is the class probability of the corresponding class.[8] For example, a class array for the classes screwdriver and wrench is of shape 2. Index 0 corresponds to screwdriver and index 1 corresponds to wrench.

To classify correctly, neural networks need to be trained first.[9] This paper trains and evaluates neural networks on the TIC Dataset. The TIC Dataset is constructed by this paper and comprises 20,400 tool images of six classes. The classes are listed below.

- drill
- hammer
- pliers
- saw
- screwdriver
- wrench

Each class comprises 3,400 tool images. A tool image is a close-up image of exactly one tool. For a class, tool images display different tools of the same class from arbitrary angles and with arbitrary backgrounds.

## 1.3 Scope

Summarizing, the scope of this paper is to determine the best-performing neural network for tool image classification. On that account, this paper focuses on neural networks for image classification. Time and resources of this paper are limited. For this reason, the neural networks are trained exclusively supervised without auxiliaries. On that account, the following approaches and techniques are excluded. Metalearning is excluded. Non-neural networks are excluded. Other computer vision tasks are excluded. Unsupervised learning and semi-supervised learning are excluded. Auxiliaries

[7] LeCun, Bengio, and G. Hinton 2015b; Lecun et al. 1998.
[8] El-Amir and Hamdy 2020.
[9] El-Amir and Hamdy 2020.

used to improve the learning of neural networks are excluded. Metalearning is the process of learning to learn, for example, learning hyperparameters of a neural network such as the architecture of the network using an evolutionary algorithm.[10] This paper regards several approaches as non-neural networks. These approaches are neural networks augmented with other machine learning algorithms, other machine learning algorithms in general or other image classification methods. Computer vision tasks other than image classification are excluded because they are less related to the problem investigated in this paper than image classification. Semi-supervised or unsupervised learning is learning without knowing all or any desired output.[11] This paper regards transfer learning, adversarial training, data augmentation, input normalization, weight decay, and multi-task learning as learning auxiliaries. Transfer learning aims to help learning a specific task by learning another task. For example, learning classification of geometric shapes in images might help learning tool image classification.[12] Adversarial training is training neural networks on adversarial examples. Adversarial examples are imperceptibly, non-randomly perturbed images. The perturbation arbitrarily changes the prediction of the neural network.[13] Data augmentation is the creation of additional training data from existing training data, for example, creating additional training images by rotating them by a random amount.[14] Input normalization improves training by re-scaling all input data into the same scale.[15] Weight decay improves training by penalizing large weights.[16] Mulit-task learning allows neural networks to learn multiple objectives. Hence, one loss function per objective is optimized during training.[17] For example, Sabour, Frosst, and G. E. Hinton 2017 made their neural network learn image classification and reconstruction of the original input.

## 1.4 Approach

This paper seeks to determine the best-performing neural network for tool image classification. The best-performing neural network for tool image classification is determined in the course of an experiment. The experiment trains and evaluates state-of-the-art neural networks for image classification on the TIC Dataset. The evaluation is based on a metric. The metric is determined based on the metrics used by image classification leaderboards. The state-of-the-art neural networks are determined in the

[10] Schaul and Schmidhuber 2010.
[11] El-Amir and Hamdy 2020.
[12] Pan and Q. Yang 2010.
[13] Szegedy, Zaremba, et al. 2014.
[14] El-Amir and Hamdy 2020.
[15] El-Amir and Hamdy 2020.
[16] El-Amir and Hamdy 2020.
[17] Caruana 1997.

course of a literature review conducted by this paper. The TIC Dataset is constructed in the course of this paper.

## 1.5 Structure

The following chapters of this paper are structured as follows.

- Chapter 2 defines and illustrates terms required to understand this paper.
- Chapter 3 defines the methodology followed to determine the best-performing neural network for tool image classification.
- Chapter 4 reports the results of the literature review on the state of the art of image classification conducted by this paper.
- Chapter 5 reports the results of the experiment conducted by this paper.
- Chapter 6 discusses the results of the experiment conducted by this paper. Furthermore, the work of this paper is placed in the state of the art of image classification, and the limitations of this work are reflected. Finally, future work and practical implications are proposed.
- Chapter 7 summarizes the contributions of this paper.

# 2 Fundamentals

This chapter defines and illustrates terms required to understand this paper.

## 2.1 Rectified Linear Unit

The Rectified Linear Unit (ReLU) is a commonly used activation function. Given an input $X$, ReLU is defined by Equation (2.1).[18]

$$relu(X) = max(0, X) \tag{2.1}$$

## 2.2 Swish Activation Function

The swish activation function *swish* is defined by Equation (2.2).[19]

$$\begin{array}{rcl} swish(x) & = & x \cdot sigmoid(x) \\ sigmoid(x) & = & \frac{1}{1+e^{-x}} \end{array} \tag{2.2}$$

## 2.3 Categorical Cross Entropy Loss Function

Given the predicted output $\hat{y}$ and the desired output $y$, the categorical cross entropy loss function $\mathcal{L}$ is defined by Equation (2.3).[20]

$$\mathcal{L}(y, \hat{y}) = \sum_{j}^{m} \sum_{y}^{n} (y_{ij} \cdot log(\hat{y}_{ji})) \tag{2.3}$$

[18] El-Amir and Hamdy 2020.
[19] Ramachandran, Zoph, and Q. V. Le 2017; Elfwing, Uchibe, and Doya 2018.
[20] El-Amir and Hamdy 2020.

## 2.4 Array

An array is a finite, ordered list of similar objects. These objects can be arrays themselves. An array nested inside another array is called subarray. All subarrays inside an array have the same structure. Accordingly, an array can be seen as ordered, finite, nested lists of objects. The objects are accessed by their position in the list called index. The index ranges from 0 to size−1. The size is the number of objects inside the array. Accessing an object is denoted by square brackets. For example, the object $a[2]$ at index 2 in the array $a = [a, b, c, d]$ is $c$.[21] Another way to denote accessing an object of a multidimensional array is to separate the indices of different dimensions by a comma. Given a multidimensional array $a$, an object at the indices $i_1, i_2, \ldots, i_n$ is denoted $a[i_1, i_2, \ldots, i_n]$ with $i_d$ being the index in the $d$th dimension, $n \in [1; D]$, and $D$ being the number of dimensions of $a$. For example, given the array $a$ of Table 2.1, the object $a[0, 0] = a[0][0]$ is $[1, 2, 3]$.[22]

An array can be described by its dimensionality and shape. The dimensionality or number of dimensions is the nesting depth of an array. The shape of an array is denoted by its size × the shape of its subarrays. If the array contains no subarrays its shape is denoted by its size. Thus, the product of all sizes in the shape is the total number of objects contained in an array.[23] This is illustrated in Table 2.1.

Table 2.1: Array

| Array | Contents | Shape | Dimensionality |
|---|---|---|---|
| $a$ | $\begin{bmatrix}\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix} & \begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix} & \begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\end{bmatrix}$ | $3 \times 3 \times 3$ | 3 |
| $a[0]$ | $\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$ | $3 \times 3$ | 2 |
| $a[0][0]$ | $\begin{bmatrix}1&2&3\end{bmatrix}$ | 3 | 1 |

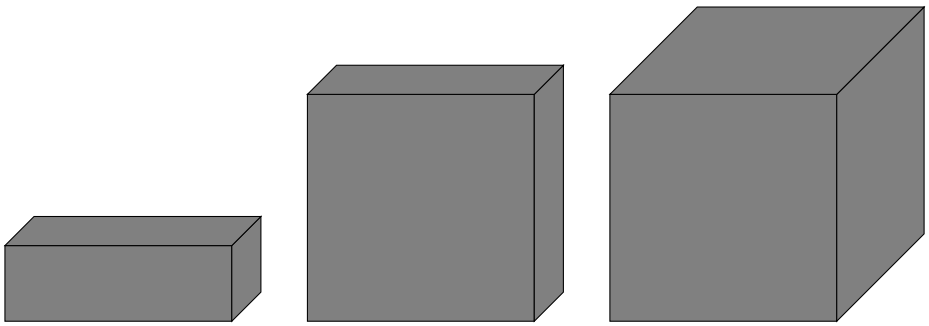
In this paper, an array is used to generalize matrices and vectors for an arbitrary number of dimensions. A vector $\vec{v}$ is seen as a one-dimensional array $v$ with each scalar $\vec{v}_i$ being the object $v[i]$ with index $i$. A matrix $A$ is seen as a two-dimensional

[21] Black 2016; Garcia and Lumsdaine 2005.
[22] de Oliveira Castro, Louise, and Barthou 2010.
[23] Black 2016; Garcia and Lumsdaine 2005.

array $a$ with each scalar $A_{ij}$ being the object $a[i][j]$ with row index $i$ and column index $j$.

### 2.4.1 Array Visualization

In the context of a Convolutional Neural Network (CNN), sometimes arrays are visualized.[24] In this paper, a three-dimensional array with shape $w \times h \times d$ is visualized as a cuboid of width $w$, height $h$, and depth $d$. A two-dimensional array with shape $w \times h$ is visualized as a cuboid or rectangle of width $w$, height $h$, and depth 1. A one-dimensional array with shape $w$ is visualized as cuboid or rectangle of width $w$, height 1, and depth 1.[25] This is illustrated in Figure 2.1.



one-dimensional two-dimensional three-dimensional

Figure 2.1: Array Visualization (own figure)

### 2.4.2 Array Slice

Given an array $a$, an array slice is denoted $a[start : end]$. $a[start : end]$ is defined as $a$ containing only the objects with indices in the range of $start$ to $end - 1$. For example, given an array $a = [0, 1, 2, 3, 4, 5]$, the array slice $a[1 : 3]$ is $[1, 2]$.[26]

[24] P. Singh and Manure 2020; El-Amir and Hamdy 2020.

[25] Ertel 2016; Rawat and Z. Wang 2017; Sabour, Frosst, and G. E. Hinton 2017; Krizhevsky, Sutskever, and G. E. Hinton 2012; LeCun, Bengio, and G. Hinton 2015b.

[26] de Oliveira Castro, Louise, and Barthou 2010.

## 2.5 Machine Learning

Machine learning in general is the art of training an algorithm to generate new information from data. The goal of machine learning is not to model explicitly how to extract this information, but to let the computer itself learn the model.[27] Machine learning has three main approaches:[28]

- Supervised learning[29]
- Unsupervised learning[30]
- Semi-supervised learning[31]

As stated in Section 1.3, this paper is focused exclusively on supervised learning.

### 2.5.1 Supervised Learning

Supervised learning is the art of training an algorithm to generate new information from data samples and associated target outputs. The target outputs can consist of numeric values or string labels. String labels can be classes or tags. The goal of supervised learning is to learn a model that can predict the correct outputs when posed with new samples. Supervised learning approaches two problems:[32]

- Classification[33]
- Regression[34]

As stated in Section 1.3, this paper is focused exclusively on classification.

### 2.5.2 Classification

Classification is the problem of assigning the correct output classes to samples. For image classification, the samples are images. For CNNs, an image is represented as a multidimensional array.[35] The output is computed using the $softmax$ function. The $softmax$ function computes the probabilities of each target class over all possible

[27] Mohri, Rostamizadeh, and Talwalkar 2012.
[28] El-Amir and Hamdy 2020.
[29] El-Amir and Hamdy 2020.
[30] El-Amir and Hamdy 2020.
[31] El-Amir and Hamdy 2020.
[32] El-Amir and Hamdy 2020.
[33] El-Amir and Hamdy 2020.
[34] El-Amir and Hamdy 2020.
[35] El-Amir and Hamdy 2020; LeCun, Bengio, and G. Hinton 2015b.

$c$ target classes. The predicted target class is the class with the highest probability. The $softmax$ function is defined by Equation (2.4).[36]

$$softmax(x) = \frac{e^x}{\sum_{i=0}^{c} e^{x_i}} \tag{2.4}$$

## 2.6 Neural Network

A neural network is a machine learning model and can be seen as a universal function approximator. A function approximator approximates a function. This function maps an input $X$ to an output $Y$.[37] In consequence, a neural network itself can be seen as a function. A neural network is structured in layers.[38] A layer receives an input and returns an output. The first layer is called input layer, the last layer is called output layer, and the remaining layers are called hidden layers.[39] A hidden layer is a layer which receives the output of its previous layer or previous layers as input. The input layer is the layer receiving the input of the neural network. The output layer is the layer returning the output of the neural network. This is illustrated in Figure 2.2. This way, layers themselves can be seen as functions. Hence, a neural network can be described as a chain of layer functions. Given the layer $F_l$ at position $l$, the input $X$, and the output $Y$ of the neural network $F$, then $Y$ is computed as defined by Equation (2.5).

$$Y = F(X) = F_l(F_{l-1} \dots (F_1(X) \dots)) \tag{2.5}$$

[36] El-Amir and Hamdy 2020.
[37] Hornik, Stinchcombe, and White 1989; Ertel 2016.
[38] Ertel 2016.
[39] LeCun, Bengio, and G. Hinton 2015b.

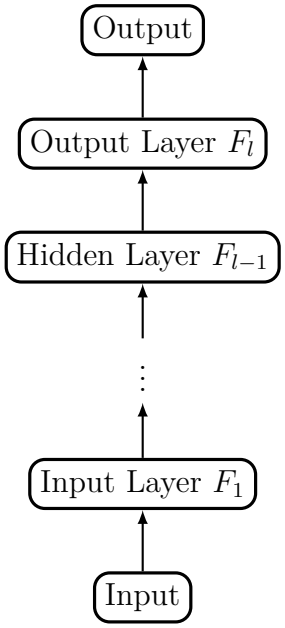



Figure 2.2: Neural Network (own figure)

A layer consists of neurons. Neurons are the basic processing units of neural networks. A neuron consists of $n$ weighted connections $w_i, i \in [1; n]$, a sum function $\sum$, and an activation function $\varphi$, see Figure 2.3. Each connection is connected to an input of the neuron. The sum function sums the weighted inputs. The activation function transforms the sum. The result of the activation function is the output $y$ of the neuron.[40] Given an input $x$, $y$ is computed as defined by Equation (2.6).

$$y = \varphi(\sum x \cdot w) \tag{2.6}$$

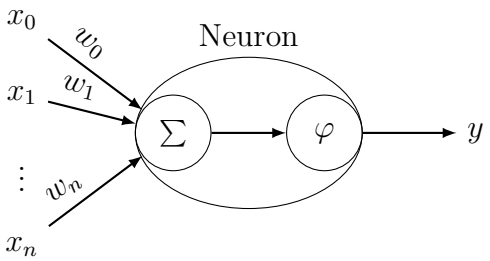



Figure 2.3: Neuron (own figure)

[40] Ertel 2016.

## 2.7 Training Neural Networks

Recapitulating, a neural network can be seen as a universal function approximator.[41] A loss function is a metric that measures how well a neural network approximates a function. The resulting value is called loss $\mathcal{L}$ .[42] For example, if a function classifying tool images is to be approximated, the loss function measures how well the neural network classifies the tool images.

Neural networks are trained by reducing the loss. The loss is reduced by adapting the parameters $\theta$ of the neural network in such a way that $\mathcal{L}$ is minimized. To properly adjust $\theta$, backpropagation[43] is used to calculate a gradient vector $\nabla\theta$. This gradient indicates by which amount the error increases or decreases if $\theta$ is increased by a very small amount. The loss function in regard to $\theta$ can be seen as high-dimensional hilly landscape, with the negative gradient vector indicating the direction of the steepest descent in that landscape. The gradient is used to descend in that landscape to find a local or the global minimum.[44] Accordingly, training a neural network can be seen as solving an optimization problem.[45]

## 2.8 Optimizer

An optimizer is an algorithm solving an optimization problem. Optimizers used in the course of this paper are Stochastic Gradient Descent (SGD) with momentum or Nesterov momentum and Root Mean Square Propagation (RMSProp). SGD and RMSProp are based on gradient descent. Gradient descent is an algorithm minimizing a loss function $\mathcal{L}(\theta)$ in regard to the parameters $\theta$ of a neural network. The loss function is minimized by changing the parameters by the negative gradient of the loss function $\nabla_\theta\mathcal{L}(\theta)$ with regard to the parameters. Before changing the parameters, the gradient is scaled by the learning rate $\eta$. This is repeated until a local or the global minimum is reached. While descending the landscape, the learning rate can be imagined as the size of a step taken in that landscape.[46] SGD approximates the gradient $\nabla_\theta\mathcal{L}(\theta)$ for each sample or batches of samples. On that account, SGD is much faster but causes the loss function to fluctuate. The fluctuation is decreased by

[41] Ertel 2016.
[42] El-Amir and Hamdy 2020.
[43] Rumelhart, G. E. Hinton, and Williams 1986.
[44] LeCun, Bengio, and G. Hinton 2015a.
[45] El-Amir and Hamdy 2020.
[46] Ruder 2016.

increasing the batch size. The change of parameters is defined in Equation (2.7).[47]

$$\theta = \theta - \eta \nabla_\theta \mathcal{L}(\theta) \tag{2.7}$$

SGD can be augmented with momentum or Nesterov momentum. Momentum accelerates SGD by adding a fraction $\gamma$ of the previous step's gradient $v_{t-1}$ to the current step's gradient. This way, the gradient is increased for dimensions whose previous gradients point into the same directions and reduced for dimensions whose previous gradients point into different directions. In consequence, a local or the global minimum is reached faster and fluctuation of the loss function is reduced. Momentum can be imagined as pushing down and gaining speed while descending the landscape. The problem is that when the landscape slopes up again the speed will result in an ascendance in the landscape. The change of parameters is defined in Equation (2.8).[48]

$$\begin{aligned} \theta &= \theta - v_t \\ v_t &= \gamma v_{t-1} + \eta \nabla_\theta \mathcal{L}(\theta) \end{aligned} \tag{2.8}$$

Nesterov momentum accelerates SGD while descending and decelerates SGD before ascending. Nesterov momentum does so by adding a fraction $\gamma$ of the previous step's gradient to an approximation of the next step's gradient $\nabla_\theta \mathcal{L}(\theta - \gamma v_{t-1})$ instead of the currents step's gradient. Approximating the next step's gradient can be imagined as looking ahead. Therefore, Nesterov momentum can be imagined as pushing down and gaining speed while looking ahead to decelerate when the landscape is about to slope up. The change of parameters is defined in Equation (2.9).[49]

$$\begin{aligned} \theta &= \theta - v_t \\ v_t &= \gamma v_{t-1} + \eta \nabla_\theta \mathcal{L}(\theta - \gamma v_{t-1}) \end{aligned} \tag{2.9}$$

RMSProp decreases fluctuation of the loss function by adapting the learning rate for each parameter $\theta_i$. The gradient of the loss function with regard to the parameter $\theta_i$ at step $t$ is called $g_{t,i}$. The learning rate is adapted by dividing it by a root mean square variation $RMS(g_t)$ of $g_t$. The root mean square variation is defined in Equation (2.10).[50]

$$RMS(g_t) = \sqrt{\gamma E[g^2]_{t-1} + (1-\gamma) g_t^2} \tag{2.10}$$

Dividing the learning rate by the root mean square variation increases the learning rate for small gradients and decreases the learning rate for large gradients. Thus, fluctuation of the loss function is decreased. The change of parameters is defined in Equation (2.11).[51]

$$\theta_{t+1} = \theta_t - \frac{\eta}{RMS(g_t) + \epsilon} g_t \tag{2.11}$$

[47] Ruder 2016.
[48] Ruder 2016.
[49] Ruder 2016.
[50] Ruder 2016.
[51] Ruder 2016.

The noise parameter $\epsilon$ is close to 0. Adding $\epsilon$ ensures that $\eta$ is never divided by zero.[52] Note that some papers refer to the fraction $\gamma$ as momentum.[53]

## 2.9 Layers

This section lists and explains basic layers used by neural networks described in this paper.

### 2.9.1 Pooling

A pooling function replaces the output at a certain location with a summary statistic of the nearby outputs.[54] The following pooling functions are used by neural networks described in this paper.

- Max Pooling
- Average Pooling
- Global Average Pooling

#### Max Pooling

The most widely used pooling technique is max pooling. Max pooling takes a maximum from all pools of input channels of the layer's input.[55] The pools can be seen as array slices. These array slices are determined by pooling size $p$ and stride $stride$. The pooling size determines the shape of the pool. The pool is shaped $p \times p$. Stride is the number of rows and columns by which the pool is shifted to determine the next pool. This shifting can be imagined as sliding the pool over the input channel. Given an input channel $X$ with size $w \times h$, max pooling $maxpooling$ is defined as described by Equation (2.12).[56]

$$\begin{aligned} &maxpooling = max(X[i : i + p, j : j + p])| \\ &i \in \{x | x_0 = 0, x \leq w - p, x_{n+1} = x_n + stride\}, \\ &j \in \{x | x_0 = 0, x \leq h - p, _{n+1} = x_n + stride\} \end{aligned} \tag{2.12}$$

[52] Ruder 2016.
[53] Simonyan and Andrew Zisserman 2014; K. He et al. 2016; S. Xie et al. 2017; G. Huang et al. 2017; Tan and Le V 2019.
[54] Goodfellow, Bengio, and Courville 2016.
[55] P. Singh and Manure 2020.
[56] Michelucci 2019.

For a better understanding, an example of max pooling is illustrated in Figure 2.4. The figure displays an example of max pooling an input channel of shape $4 \times 4$, with a stride of 2, and a pooling size of 2. The different pools are highlighted in different colors. The result of a pool is highlighted in the color of that pool.

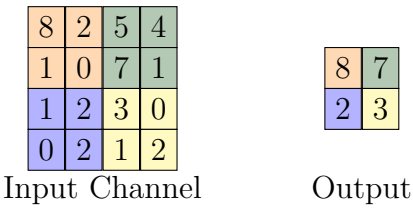



Figure 2.4: Max Pooling Illustration (own figure)

### Average Pooling

Another pooling technique is average pooling. Average pooling is used by DenseNet-264[57] which is described in this paper. Understanding this neural network requires understanding average pooling. Average pooling works exactly the same as max pooling except that it takes the average instead of the maximum. Given an input channel $X$ with size $w \times h$, average pooling $avgpooling$ is defined as described by Equation (2.13).[58]

$$\begin{aligned} avgpooling &= average(X[i : i + p, j : j + p])| \\ average(X) &= \frac{\sum X}{|X|} \end{aligned} \tag{2.13}$$

### Global Average Pooling

Another pooling technique is global average pooling. Global average pooling is used by ResNet-152,[59] ResNeXt-101,[60] DenseNet-264,[61] and EfficientNet-B7[62] which are described in this paper. Understanding these neural networks requires understanding global average pooling. Global average pooling takes the average of each input channel. The input is an array of input channels. The output is an array containing the averages.

[57] G. Huang et al. 2017.
[58] Michelucci 2019.
[59] K. He et al. 2016.
[60] S. Xie et al. 2017.
[61] G. Huang et al. 2017.
[62] Tan and Le V 2019.

Given an array of $d$ input channels $X$, global average pooling *globalavgpooling* is defined as described by Equation (2.14).[63]

$$\begin{aligned} globalavgpooling(X) &= concat(globalavgpooling_0, \ldots, globalavgpooling_{d-1}) \\ globalavgpoolingg_i &= average(X_i) \\ average(X) &= \frac{\sum X}{|X|} \end{aligned} \tag{2.14}$$

### 2.9.2 Dense

A dense layer is a layer in which each neuron is connected to each input of the layer, see Figure 2.5. Given the weights of the neurons $W$, an input $X$, and an activation function $\varphi$, the dense layer *dense* is described by Equation (2.15).[64]

$$dense(X) = \varphi(X \cdot W) \tag{2.15}$$

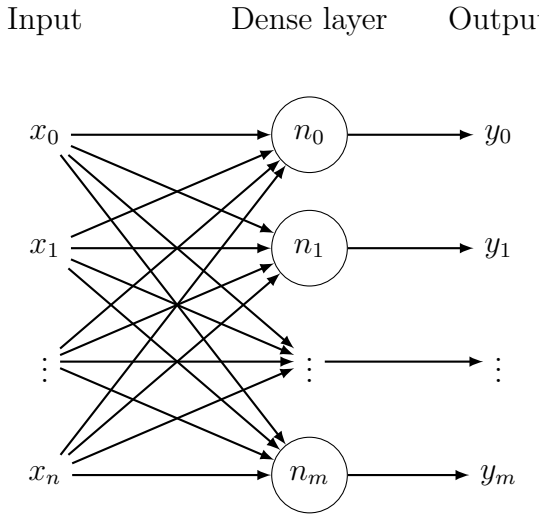



Figure 2.5: Dense Layer (own figure)

### 2.9.3 Batch Normalization

Batch normalization is a technique to speed up training. Training is sped up by smoothing the optimization landscape and stabilizing the gradients.[65] Batch normalization normalizes an input by subtracting the mean and dividing by the standard

[63] Lin, Q. Chen, and S. Yan 2013.
[64] P. Singh and Manure 2020.
[65] Santurkar et al. 2018.

deviation. Mean and standard deviation are approximated batch-wise. Computation over a batch is more efficient, but mean and standard deviation vary dependent on the batch. Therefore, the true mean and standard deviation are approximated over multiple batches by learnable parameters. Given a batch of $m$ inputs $\mathcal{B} = \{x_1, x_2, \ldots, x_m\}$, learnable parameters $\gamma, \beta$, and a noise $\epsilon$, batch normalization *batchnorm* is defined by Equation (2.16).[66] Noise ensures that division by 0 does not occur.

$$\begin{array}{lcl} batchnorm(x) & = & \gamma\hat{x} + \beta \\ \hat{x} & = & \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} \\ \mu & = & \frac{1}{m}\sum_{i=1}^{m} b_i | b_i \in \mathcal{B} \\ \sigma & = & \frac{1}{m}\sum_{i=1}^{m}(b_i-\mu)^2 | b_i \in \mathcal{B} \end{array} \tag{2.16}$$

### 2.9.4 Dropout

Dropout is a regularization technique addressing the problem of overfitting. Dropout randomly drops neurons during training. Dropping means removing the neuron temporarily, along with its connections. The probability of a neuron being dropped is called dropout rate $q$. Neurons are only dropped during training. After training, each neuron is retained and their weights are scaled down by the probability of a neuron being retained $p = 1 - q$.[67]

[66] Ioffe and Szegedy 2015.
[67] Srivastava et al. 2014.

# 3 Methodology

This paper seeks to determine the best-performing neural network for tool image classification. In the course of the literature review conducted by this paper, no paper about neural networks for tool image classification was found. On that account, the neural network is determined based on the state of the art of neural networks for image classification. This chapter defines the methodology to determine this neural network. To determine this neural network the state-of-the-art neural networks need to be compared in regard to their performance. Therefore, a metric, a dataset, and the state-of-the-art neural networks are required. The metric operationalizes performance. Thus, performance is comparable. The metric is selected and defined in Section 3.1. The dataset is required to train and evaluate the neural networks.[68] The methodology of the dataset construction is defined in Section 3.2. This paper focuses on the state of the art of neural networks for image classification, see Section 1.3. The state of the art is analyzed in the course of a literature review conducted by this paper. The methodology of the literature review is defined in Section 3.3. The best-performing neural network for tool image classification is determined in the course of an experiment. This experiment is described in Section 3.4.

## 3.1 Metric

Performance of neural networks for image classification is measured on benchmark datasets. The benchmark datasets are listed in Section A.2. For these datasets, performance is measured in classification accuracy or error.[69] Classification accuracy and error are equivalent.[70] Based on these datasets, this paper measures performance in classification accuracy. For reasons of simplification, classification accuracy is referred to as accuracy in the context of this paper. Accuracy is a metric to measure the classification performance of a classifier. Accuracy *acc* is defined as the amount of correctly predicted labels *correct* to the total amount of predictions *total*, see Equation (3.1).[71]

[68] LeCun, Bengio, and G. Hinton 2015a.

[69] Deng et al. 2009; Krizhevsky 2012; LeCun, Cortes, and Burges 2010; Netzer et al. 2011; Z. Liu et al. 2016; H. Xiao, Rasul, and Vollgraf 2017; Darlow et al. 2018; Bossard, Guillaumin, and Van Gool 2014; van Horn et al. 2018; Krause et al. 2013; Cohen et al. 2017; Clanuwat et al. 2018; Wah et al. 2011; Sabour, Frosst, and G. E. Hinton 2017; Combalia et al. 2019; Codella et al. 2018; Tschandl, Rosendahl, and Kittler 2018.

[70] Bansal and R. Mahajan 2019.

[71] Bansal and R. Mahajan 2019.

$$acc = \frac{correct}{total} \tag{3.1}$$

As a result, accuracy can be misleading for an imbalanced dataset. For example, given a dataset consisting of 99% samples labeled $A$ and 1% samples labeled $B$, a classifier purely predicting $A$ gets an accuracy of 99%. Consequently, this classifier achieves a high accuracy despite predicting all samples labeled $B$ wrong. The dataset constructed in the course of this paper is balanced. Hence, accuracy is suitable for this dataset.

## 3.2 Dataset Construction

This paper seeks to determine the best-performing neural network for tool image classification. For this reason, optimally, this paper would construct a huge dataset with various images from different classes of tools.[72] However, time and resources of this paper are limited. Labeling huge amounts of data, such as data mined from the web, requires resources. On that account, this paper exclusively uses labeled data to construct the dataset. Labeled data can be acquired from already existing datasets and from creating inherently labeled images.

This paper creates inherently labeled images. Inherently labeled images are created by taking all images of one class before taking images of another class and storing them in different folders. The resulting folder structure is comprised by a root folder containing one folder for each image class. The classes were chosen based on the available tools. The resulting classes are listed below.

- drill
- hammer
- pliers
- saw
- screwdriver
- wrench

The dataset is constructed for a tool image classification task. Thus, the created images each consist of exactly one tool. In real world scenarios, tools are presented from various angles and with various backgrounds. Due to this, the created images are taken from arbitrarily chosen angles and with arbitrarily chosen backgrounds.

[72] Howard 2013; Krizhevsky 2012; Deng et al. 2009.

A local joinery provided tools and the location for the creation of the dataset free of charge. The dataset was created by placing a tool in an arbitrarily chosen position and taking a series of close-up images. During the series, the camera was moved around the tool to create arbitrary angles. For the next series, the tool, the background, and/or the position of the tool was changed. The images were created by volunteers and the author of this paper: Nina Eichler, Sophia Faißt, Manuel Krumbacher, and Fabian Wolf. The resulting dataset is provided publicly available under the Creative Commons Attribution-ShareAlike 4.0 International License. The dataset is appended in the digital appendix, see Section A.6. The dataset is described in Section 3.4

## 3.3 Literature Review

The literature review consists of a preparation and an implementation phase. In the preparation phase, literature inclusion and exclusion criteria, a literature source list, and a term table are constructed. The inclusion and exclusion criteria determine whether or not a paper is selected for the literature review. The inclusion and exclusion criteria are listed in Section A.3. The literature source list lists all sources of papers. These sources are journals, conference proceedings, full-text databases, image classification leaderboards, and scientific search engines. The literature source list is displayed in Section A.4. The term table lists terms related to the problem statement described in Section 1.2. As a result, the term table contains all terms that can be used to derive search queries. The term table is displayed in Section A.5. In the implementation phase, the sources are systematically screened using the search queries. The resulting papers are selected using the inclusion and exclusion criteria. Following Webster and Watson 2002, the selected papers are analyzed in Concept Matrix A.1. The concept matrix maps each paper to concepts of neural networks they support.

The state-of-the-art neural networks for image classification are selected based on these concepts. For each concept this paper determines the best-performing neural network for image classification. Performance of neural networks for image classification is measured on benchmark datasets. The benchmark datasets are listed in Section A.2. For these datasets, neural networks are ranked on leaderboards based on accuracy. The accuracy differs for the same neural network on different leaderboards. Accordingly, different leaderboards have differently complex tasks. Thus, neural networks cannot be compared by averaging accuracy across all leaderboards. Furthermore, not every neural network is listed on every leaderboard.[73] Hence, the neural networks are

[73] Deng et al. 2009; Krizhevsky 2012; LeCun, Cortes, and Burges 2010; Netzer et al. 2011; Z. Liu et al. 2016; H. Xiao, Rasul, and Vollgraf 2017; Darlow et al. 2018; Bossard, Guillaumin, and Van Gool 2014; van Horn et al. 2018; Krause et al. 2013; Cohen et al. 2017; Clanuwat et al. 2018; Wah et al. 2011; Sabour, Frosst, and G. E. Hinton 2017; Combalia et al. 2019; Codella et al. 2018; Tschandl, Rosendahl, and Kittler 2018.

not comparable by averaging rank across all leaderboards. Accordingly, the neural networks are compared pairwise. A neural network $A$ is ranked above a neural network $B$ if $A$ performs better on a larger number of leaderboards that contain both $A$ and $B$.

Neural networks ranked on the leaderboards use different auxiliaries to further improve performance.[74] As stated in Section 1.3, this paper focuses exclusively on the neural network. Therefore, only the underlying neural network is regarded. If several versions of the underlying neural network exist, the best-performing version according to the paper originally proposing the neural network is regarded as the best-performing state-of-the-art neural network.

In summary, the result of this literature review are state-of-the-art concepts of neural networks for image classification and the best-performing state-of-the-art neural network for each concept. The results of the literature review are reported in Chapter 4.

## 3.4 Experiment

This paper seeks to determine the best-performing neural network for tool image classification. The best-performing neural network is determined in the course of an experiment. The experiment is conducted by training and evaluating selected neural networks on a dataset. The Dataset, selected neural networks, training, and evaluation are described in this chapter. To measure performance, accuracy is used, see Section 3.1. The results of this experiment are reported in Chapter 5. To conduct the experiment, Movilizer GmbH provided 200 hours on an Amazon Web Service g4dn.xlarge instance running a Deep Learning AMI (Ubuntu 18.04).[75] The g4dn.xlarge instance provides 4 vCPUs, 16GB RAM, 125GB storage, and a NVIDIA T4 GPU.[76] The experiment is implemented in Python 3[77] using Keras 2.2.4.2,[78] Tensorflow 2.1.0,[79] CUDA 10.1,[80] and cuDNN.[81] The dataset and code used to conduct the experiment are appended in the digital appendix, see Section A.6.

[74] S. Lim et al. 2019; Harris et al. 2020; Y. Huang et al. 2019; Xiao Wang et al. 2019; Q. Xie et al. 2019; Touvron et al. 2019; Darlow et al. 2018; Tan and Le V 2019.
[75] Amazon Web Services, Inc. 2020b.
[76] Amazon Web Services, Inc. 2020a.
[77] Van Rossum and Drake 2009.
[78] Chollet et al. 2015a.
[79] Martn Abadi et al. 2015.
[80] NVIDEA 2007.
[81] Chetlur et al. 2014.

### 3.4.1 Dataset

The dataset is constructed as described in Section 3.2. The constructed dataset is comprised of tool images for image classification. Therefore, it is named TIC Dataset. The TIC Dataset is comprised of 20,400 tool images of six classes. The classes are listed below.

- drill
- hammer
- pliers
- saw
- screwdriver
- wrench

Each class comprises 3,400 tool images. A tool image is a close-up image of exactly one tool. For a class, tool images display different tools of the same class from arbitrary angles and with arbitrary backgrounds. Figure 3.1 illustrates this for the class screwdriver with two images of different tools.

Figure 3.1: Two Images from the TIC Dataset of Class Screwdriver (own figure)

The dataset is split into a training, validation, and test dataset. For each class, 60% of the images are randomly split for the training dataset, 20% for the validation dataset, and 20% for the test dataset. The resulting training dataset contains 12,240, the resulting validation dataset contains 4,080, and the resulting test dataset contains 4,080 images. For each of these datasets, the images are evenly distributed over all classes. The training dataset is used to train a neural network. The validation dataset and the test dataset are not used for training. On that account, the neural network cannot overfit to the validation dataset and the test dataset. Thus, the validation dataset can be used to explore different hyperparameters and monitor whether or not

the neural network overfits to the training dataset. Based on the performance on the validation dataset, the neural network is chosen for evaluation. This way, the neural network might develop a dependency on the validation dataset. For example, a neural network might perform well on the validation dataset by chance but less well on other data. For this reason, the neural network is evaluated only once on the test dataset. The performance of the neural network on the test dataset is regarded as the performance of the neural network.[82]

### 3.4.2 Neural Network Selection

To determine the best-performing neural network for tool image classification, optimally, this paper would conduct a neural network architecture search and hyperparameter search. However, resources and time of this paper are limited. Due to that, conducting a neural network architecture search and hyperparameter search is not possible.

Tool image classification is a sub-task of image classification. Therefore, this paper selects the state-of-the-art neural networks for image classification to conduct the experiment. The state-of-the-art neural networks for image classification are determined as described in Section 3.3. For each selected neural network the network architecture and the hyperparameters are adopted from the paper originally proposing the neural network. The following neural networks are selected to conduct the experiment:

- VGG-19, see Section 4.1.[83]
- ResNet-152, see Section 4.2.[84]
- ResNext-101, see Section 4.3.[85]
- DenseNet-264, see Section 4.4.[86]
- EfficientNet-B7, see Section 4.6.[87]

This paper implements the selected neural networks based on their Keras implementation.[88] Note that CapsNet is excluded since it does not reach state-of-the-art performance on image classification except for classification of digits or letters.

[82] El-Amir and Hamdy 2020.
[83] Simonyan and Andrew Zisserman 2014.
[84] K. He et al. 2016.
[85] S. Xie et al. 2017.
[86] G. Huang et al. 2017.
[87] Tan and Le V 2019.
[88] Chollet et al. 2015b.

### 3.4.3 Training

The neural networks are trained on the training dataset and their performance is monitored on the validation dataset. Training is carried out by optimizing the categorical cross entropy loss function,[89] see Section 2.7. Each selected neural network is trained with the number of epochs, the optimizer, and the hyperparameters of the optimizer as specified in the paper originally proposing the neural network. The hyperparameters of the optimizer are initial learning rate, learning rate schedule, and momentum or Nesterove momentum if used. The learning rate schedule determines how the learning rate changes with epochs progressing. The batch size specified in the original papers are powers of 2. However, those batch sizes are too large to fit the available GPU. Therefore, the power is reduced by one until the batch size fits the available GPU. For each neural network, the training hyperparameters, namely number of epochs, the optimizer, the hyperparameters of the optimizer, and the batch size are displayed in Table 3.1.[90]

For EfficientNet-B7, the number of epochs is not specified, instead EfficientNet-B7 is trained until convergence.[91] The available 200 hours on the provided hardware might not be enough to train EfficientNet-B7 until convergence. Therefore, the other neural networks are trained before EfficientNet-B7. EfficientNet-B7 is trained for the remaining time.

Table 3.1: Training Hyperparameters[92]

| **Neural Network** | **Training Hyperparameters** | |
|---|---|---|
| VGG-19 | Epochs: | 74 |
| | Optimizer: | SGD |
| | Initial Learning Rate: | 0.01 |
| | Learning Rate Schedule: | divide learning rate by 10 on accuracy plateaus. |
| | Momentum: | 0.9 |
| | Batch Size: | 32 |
| ResNet-152 | Epochs: | 120 |
| | Optimizer: | SGD |
| | Initial Learning Rate: | 0.1 |
| | Learning Rate Schedule: | divide learning rate by 10 on accuracy plateaus. |

[89] Michelucci 2019; El-Amir and Hamdy 2020; P. Singh and Manure 2020.
[90] Simonyan and Andrew Zisserman 2014; K. He et al. 2016; S. Xie et al. 2017; G. Huang et al. 2017; Tan and Le V 2019.
[91] Tan and Le V 2019.

Table 3.1: Training Hyperparameters[92]

| Neural Network | Training Hyperparameters | |
|---|---|---|
| | Momentum: | 0.9 |
| | Batch Size: | 32 |
| ResNeXt-101 | Epochs: | 120 |
| | Optimizer: | SGD |
| | Initial Learning Rate: | 0.1 |
| | Learning Rate Schedule: | divide learning rate by 10 at epoch 30, 60, and 90. |
| | Momentum: | 0.9 |
| | Batch Size: | 16 |
| DenseNet-264 | Epochs: | 90 |
| | Optimizer: | SGD |
| | Initial Learning Rate: | 0.1 |
| | Learning Rate Schedule: | divide learning rate by 10 at epoch 30 and 60. |
| | Nesterov Momentum: | 0.9 |
| | Batch Size: | 32 |
| EfficientNet-B7 | Epochs: | not specified |
| | Optimizer: | RMSProp |
| | Initial Learning Rate: | 0.256 |
| | Learning Rate Schedule: | decay learning rate by 0.97 each 2.4 epochs. |
| | Momentum: | 0.9 |
| | Batch Size: | 1 |

Note that metalearning, semi- or unsupervised learning, transfer learning, adversarial training, data augmentation, input normalization, weight decay, and multi-task learning are excluded from the scope of this paper, see Section 1.3. In consequence, they are excluded from the experiment even if they are used in the original papers.

[92] Simonyan and Andrew Zisserman 2014; K. He et al. 2016; S. Xie et al. 2017; G. Huang et al. 2017; Tan and Le V 2019.

### 3.4.4 Evaluation

As mentioned, during training the performance of each neural network is monitored on the validation dataset. The performance is measured at the end of each epoch. After training for the entire number of epochs, the neural network from the epoch with the highest performance is evaluated. A neural network is evaluated on the test dataset. The resulting performance measured in accuracy is reported in Chapter 5.

# 4 State of the Art of Image Classification

This chapter reports the results of the literature review on state of the art of image classification described in Section 3.3. Each Section of this chapter describes one concept and best-performing neural network implementing that concept. All concepts are variations of CNNs.

## 4.1 Convolutional Neural Network

CNNs are neural networks using convolutional layers. Convolutional layers are designed to input multidimensional arrays and output multidimensional arrays. For image classification, the input is composed of one or more two-dimensional arrays called channels. The channels share the same shape $w_{in} \times h_{in}$. Consequently, the input can be represented as a three-dimensional array of shape $w_{in} \times h_{in} \times d_{in}$ with $d_{in}$ being the number of channels. The output is composed of one or more two-dimensional arrays called feature maps. The feature maps share the same shape $w_{out} \times h_{out}$. Consequently, the output can be represented as a three-dimensional array of shape $w_{out} \times h_{out} \times d_{out}$ with $d_{out}$ being the number of feature maps.[93] This is illustrated in Figure 4.1.

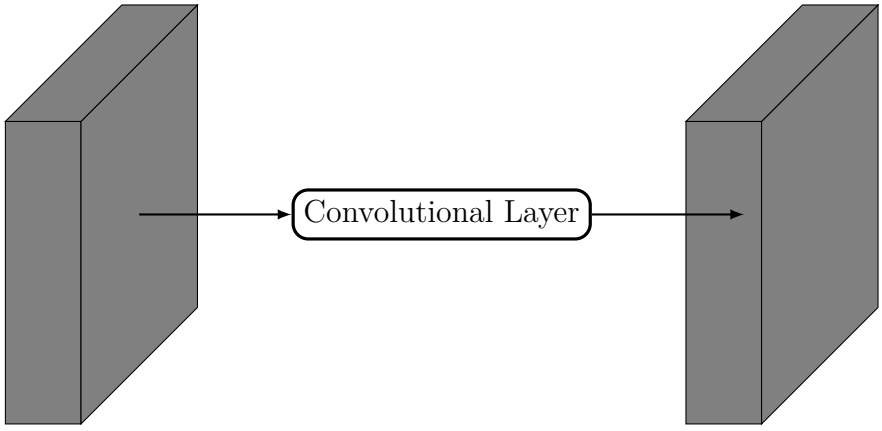



Figure 4.1: Convolutional Layer (own figure)

[93] LeCun, Bengio, and G. Hinton 2015b; Lecun et al. 1998.

Convolutional layers consist of neurons which only connect to a small region of the layer's input, see Figure 4.2. This region is called receptive field.[94] Within a receptive field, each connection of a neuron is connected to one element of the input. Consequently, a receptive field can be seen as an array slice of the layer's input. This array is of shape $k \times k \times d$. The set of weights of a convolutional neuron's connections is referred to as filter or kernel. The kernel can be represented as a three-dimensional array of shape $k \times k \times d$. $k$ is the size of the kernel.[95]

A convolutional layer holds one neuron for each receptive field and for each kernel. This means, neurons of the same kernel share the same weights. On that account, neurons of the same kernel can be seen as $r$ copies of a neuron connected to all receptive fields or applying the same neuron to each receptive field. Applying the same neuron to each receptive field can be imagined as sliding the neuron over the input. The output of a neuron is computed as described in Section 2.6. Given the input $X$, a kernel $K$, and the receptive field $X[i : i+k, j : j+k]$, the output of a neuron $y$ is computed as defined by Equation (4.1). As a result, the output of all neurons of a kernel is computed by computing the output for all receptive fields with $i \in \{x|x_0 = 0, x \leq w - k, x_{n+1} = x_n + stride\}$ and $j \in \{x|x_0 = 0, x \leq h - k,_{n+1} = x_n + stride\}$. This computation loosely relates to discrete convolution. For this reason, CNNs are called convolutional. The resulting output is a feature map. Hence, the output of a convolutional layer is computed by computing all feature maps. Derived from this, the shape of the output is $\frac{w-k+2 \cdot padding}{stride} + 1 \times \frac{h-k+2 \cdot padding}{stride} + 1 \times k$. Padding can be described as increasing the input's width and height by *padding* on both sides by filling with zeros. Imagining sliding the neuron over the input, *stride* is the step size by which the neuron is slid.[96]

$$y = \varphi(\sum X[i : i + k, j : j + k] \cdot K) \tag{4.1}$$

[94] Lecun et al. 1998.
[95] Rawat and Z. Wang 2017; LeCun, Bengio, and G. Hinton 2015b.
[96] Goodfellow, Bengio, and Courville 2016.

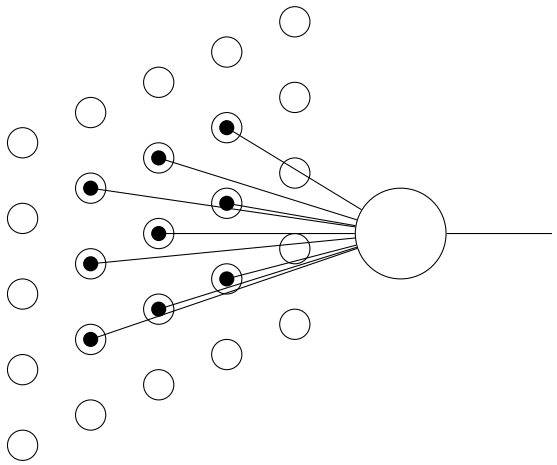


Figure 4.2: Neuron of a Convolutional Neural Network (own figure)

For better understanding, convolution, stride, and padding are illustrated in the following paragraphs.

**Convolution** For example, consider the case of a grayscale image represented as one channel $X$, a kernel $K$, and an identity activation function $\varphi$. Then, the output of a convolutional neuron is computed by laying the kernel on top of the channel. Each value of the activation map is computed by multiplying the overlapping values. The resulting values are summed. The sum is processed by the activation function. In this case, the activation function is the identity function. Thus, the result is the sum itself. The activation map $Y$ is computed by sliding the kernel over each row and each column by the factor of stride. This is illustrated in Figure 4.3.[97]

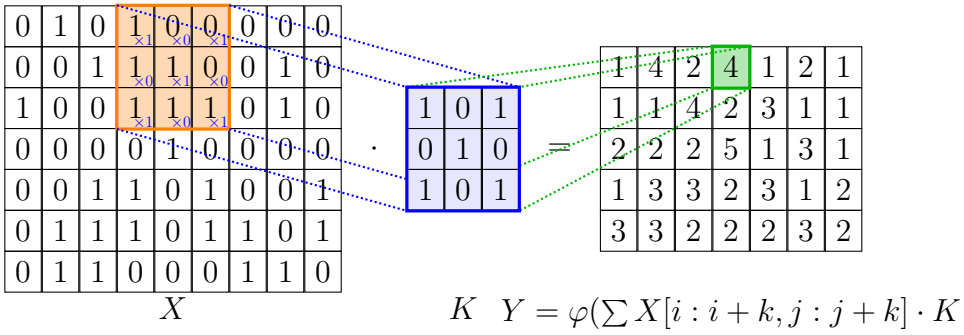



Figure 4.3: Convolution Illustration (own figure)

[97] Goodfellow, Bengio, and Courville 2016.

**Stride** For example, consider the case of a grayscale image represented as one channel $X$, a kernel $K$, and a *stride* of 1 and 2 respectively. A stride of 1 means sliding the kernel with a step size of 1. A stride of 2 means sliding the kernel with a step size of 2. This is illustrated in Figure 4.4.[98]

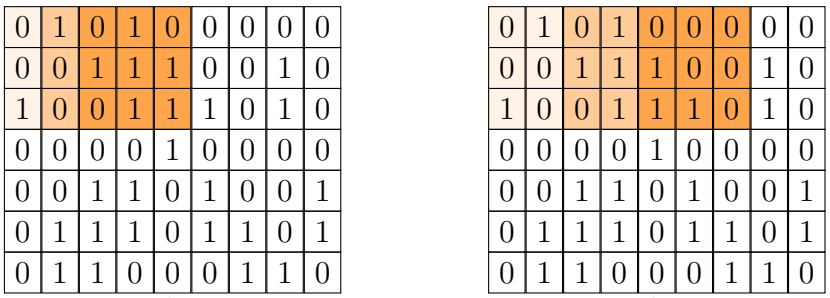



Figure 4.4: Stride Illustration (own figure)

**Padding** For example, consider the case of a grayscale image represented as one channel $X$ and a *padding* of 1. A padding of 1 adds 1 row and column of 0s respectively to each edge of the channel. This is illustrated in Figure 4.5.[99]

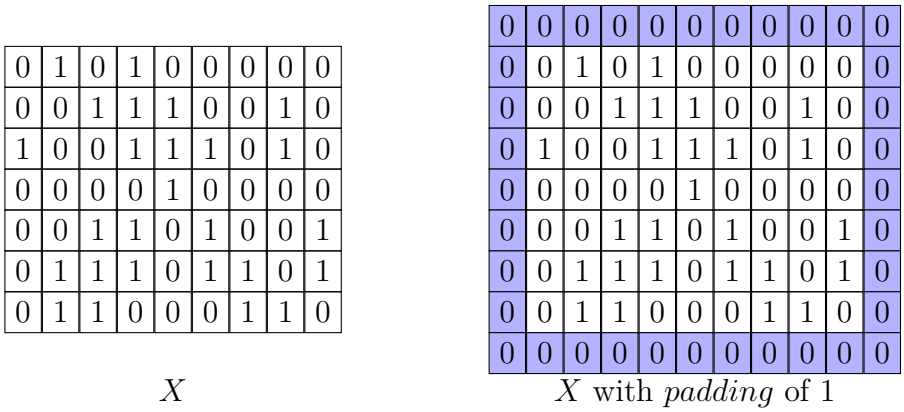



Figure 4.5: Padding Illustration (own figure)

The best-performing CNN found in the course of the literature review is the 19-layer version of Simonyan and Andrew Zisserman 2014's VGG, called VGG-19. The input of VGG-19 is a 224-by-224-pixel, Red, Green, and Blue (RGB) image. The output of VGG-19 comprises the probabilities of the $c$ target classes. VGG-19 is comprised

[98] Goodfellow, Bengio, and Courville 2016.
[99] Goodfellow, Bengio, and Courville 2016.

of 16 convolutional layers and 3 dense layers. All layers except the output layer use the ReLU activation function. The output layer uses the softmax activation function. The convolutional layers have different numbers of kernels. The kernels are of kernel size 3. Max pooling is applied after layer number 2, 4, 8, 12, and 16. The pooling size is 2 and the pooling stride is 2. For each layer, the number of kernels and neurons respectively are outlined in Table 4.1.[100]

Table 4.1: VGG-19 Configuration

| **Layer Number** | **Layer** | **Number of Neurons/Kernels** |
|---|---|---|
| 1 | Convolutional Layer | 64 |
| 2 | Convolutional Layer | 64 |
| 3 | Convolutional Layer | 128 |
| 4 | Convolutional Layer | 128 |
| 5 | Convolutional Layer | 256 |
| 6 | Convolutional Layer | 256 |
| 7 | Convolutional Layer | 256 |
| 8 | Convolutional Layer | 256 |
| 9 | Convolutional Layer | 512 |
| 10 | Convolutional Layer | 512 |
| 11 | Convolutional Layer | 512 |
| 12 | Convolutional Layer | 512 |
| 13 | Convolutional Layer | 512 |
| 14 | Convolutional Layer | 512 |
| 15 | Convolutional Layer | 512 |
| 16 | Convolutional Layer | 512 |
| 17 | Dense Layer | 4096 |
| 18 | Dense Layer | 4096 |
| 19 | Dense Layer | $c$ |

[100]Simonyan and Andrew Zisserman 2014.

## 4.2 Residual Network

Residual networks address the degradation of accuracy of deep neural networks. Deep neural networks saturate, then degrade rapidly with increasing depth. Degradation is not caused by overfitting as increasing depth of suitably deep neural networks increases training error.[101]

> The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize. Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it. There exists a solution by construction to the deeper model: the added layers are identity mappings, and the other layers are copied from the learned shallower model. The existence of this constructed solution indicates that a deeper model should produce no higher training error than its shallower counterpart. (K. He et al. 2016)

Residual networks address the degradation problem by letting stacked layers learn the residual function $\mathcal{F}(X) = \mathcal{H}(X) - X$ instead of the desired function $\mathcal{H}(X)$. Thus, the original desired function is transformed into $\mathcal{F}(X) + X$. The shape of $\mathcal{F}(X)$ must match $X$. For differing shapes, $X$ is projected by $W$ to match shapes. This results in $\mathcal{F}(X) + WX$. The formulation of $\mathcal{F}(X) + X$ can be implemented with shortcut connections. Shortcut connections are connections that skip one or more layers. A stack of layers that is skipped by a shortcut connection is called residual block, see Figure 4.6. This is based on the hypothesis that stacked layers can learn any function. This hypothesis is equivalent to the hypothesis that stacked layers can learn the residual functions $\mathcal{H}(X) - X$. [102] Both approaches should be able to learn the desired functions. The difficulty of learning may vary. Using residual networks, deeper neural networks can be learned.[103]

[101] K. He et al. 2016.
[102] assuming that the input and output are of the same dimensions
[103] K. He et al. 2016.

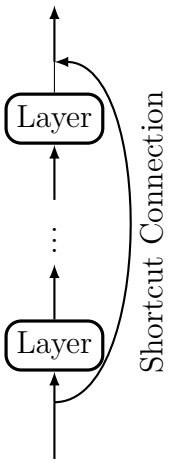



Figure 4.6: Residual Block (own figure)

The best-performing residual network found in the course of the literature review is the 152-layer version of K. He et al. 2016's ResNet, called ResNet-152. The input of ResNet-152 is a 224-by-224-pixel, RGB image. The output of ResNet-152 comprises the probabilities of the $c$ target classes.[104]

ResNet-152 is comprised of 151 convolutional layers followed by 1 dense layer. The convolutional layers have a stride of 1, unless specified otherwise. The convolutional layers use batch normalization and the ReLU activation function. Batch normalization is applied after every convolution before the activation function. The dense layer has $c$ neurons and uses the softmax activation function. The first convolutional layer is followed by max pooling, has a kernel size of 7, and a stride of 2. Max pooling is applied with a pooling size of 3 and a pooling stride of 2.[105]

The remaining covolutional layers are arranged in residual blocks. A residual block consists of three stacked convolutional layers and a shortcut connection. The first and third layer have a kernel size of 1. The second layer has a kernel size of 3. The first and third layer are used to reduce, then increase (restore) dimensions. This way, the second layer becomes a bottleneck with smaller input/output dimensions. On that account, computation costs are reduced. The shortcut connection skips the three layers. The shortcut connection uses identity mapping. The output of the identity mapping is added element-wise to the output of the skipped layers. The result transformed by the ReLU activation function is the output of the block. ResNet-152 consists of 4 types of stacked blocks. The types differ only in the number of kernels of their layers. The configurations of each residual block are outlined in Table 4.2.[106]

[104]K. He et al. 2016.
[105]K. He et al. 2016.
[106]K. He et al. 2016.

The first layer of the first block of a type has a stride of 2. This way, the feature map size is halved while the number of filters is doubled. Hence, the time complexity per layer is preserved. The last layer of the last block is followed by global average pooling.[107]

The whole configuration of ResNet-152 is outlined in Table 4.2.[108]

Table 4.2: ResNet-152 Configuration. Note that each $k \times k$ conv $K$ denotes a convolutional layer with $K$ kernels of size $k$. A residual block is denoted as an array of layers.

| **Layer/Block** | **Configuration** |
|---|---|
| Input Layer | $7 \times 7$ conv 64, stride 2, and $3 \times 3$ max pooling, stride 2 |
| Residual Block $1-3$ | $\begin{bmatrix} 1 \times 1 \text{ conv } 64 \\ 3 \times 3 \text{ conv } 64 \\ 1 \times 1 \text{ conv } 256 \end{bmatrix} \times 3$ |
| Residual Block $4-11$ | $\begin{bmatrix} 1 \times 1 \text{ conv } 128 \\ 3 \times 3 \text{ conv } 128 \\ 1 \times 1 \text{ conv } 512 \end{bmatrix} \times 8$ |
| Residual Block $12-47$ | $\begin{bmatrix} 1 \times 1 \text{ conv } 256 \\ 3 \times 3 \text{ conv } 256 \\ 1 \times 1 \text{ conv } 1024 \end{bmatrix} \times 36$ |
| Residual Block $48-50$ | $\begin{bmatrix} 1 \times 1 \text{ conv } 512 \\ 3 \times 3 \text{ conv } 512 \\ 1 \times 1 \text{ conv } 2048 \end{bmatrix} \times 3$ |
| Output Layer | global average pooling, dense with $c$ neurons |

## 4.3 Inception Network

The main idea of the inception network is to approximate an optimal local sparse network by dense components. An optimal local sparse network can be constructed layer by layer. For each layer, the correlation statistics of the last layer are clustered into groups of high correlation. These clusters form the units of the layer and are connected to the units of the previous layer. This can be imagined in terms of the Hebbian principle ("what fires together, wires together"). Neurons in a correlation

[107] K. He et al. 2016.
[108] K. He et al. 2016.

cluster ("what fires together") are connected to neurons in another correlation cluster ("wires together"). Clusters concentrated in a narrow region can be covered by a convolution of small size. Clusters concentrated in a wider region can be covered by a convolution of a bigger size.[109]

Inception networks approximate the optimal local sparse network by combining those convolutions. The convolutions are combined in inception modules. The naive version of an inception module combines max pooling and three convolutional layers with kernel size 1, 3, and 5. These are applied in parallel to the input of the inception module. This way, the inception module is branched into four distinct paths. The resulting activation maps are concatenated to one output. That way, the four paths are merged into one output. This output is the output of the inception module. The input of the inception module is an array of channels. This naive version of the inception module is illustrated in Figure 4.7.[110]

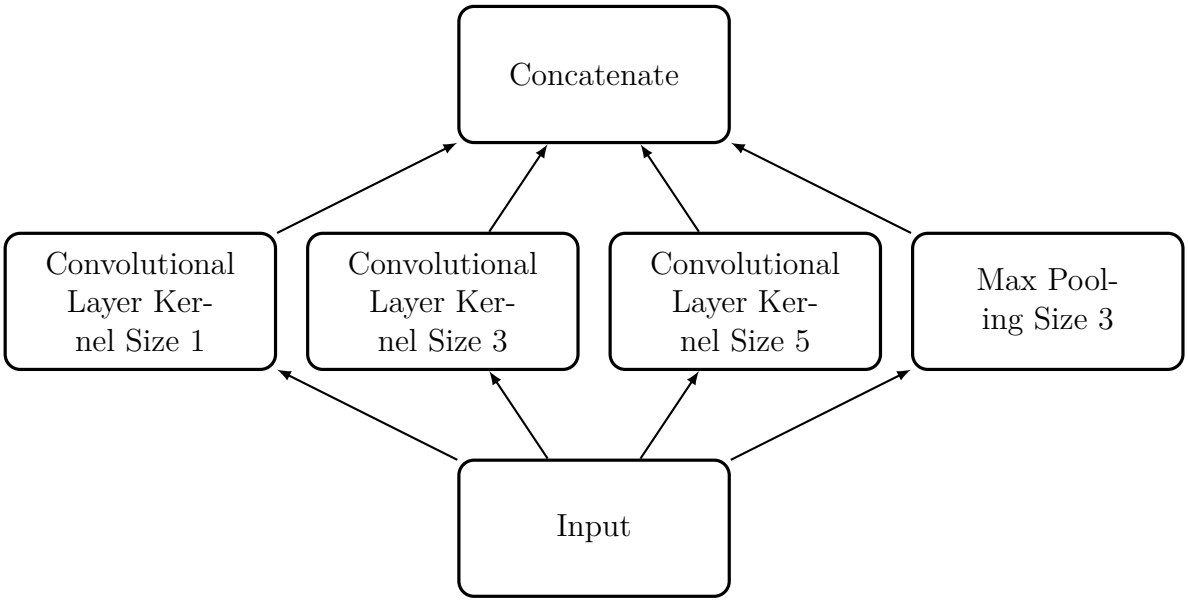



Figure 4.7: Naive Inception Module (own figure)

Inception networks are neural networks containing inception modules. The naive version of inception modules increases the number of outputs. Accordingly, stacking inception modules leads to a rapid increase of computational complexity. Thus, the number of outputs is reduced by dimension reduction. Dimension reduction is computed by convolutions of kernel size 1. This convolution is applied before the compu-

[109] Szegedy, W. Liu, et al. 2015.
[110] Szegedy, W. Liu, et al. 2015.

tationally expensive convolutions of kernel size 3 and 5 as well as after max pooling. The resulting inception module is illustrated in Figure 4.8.[111]

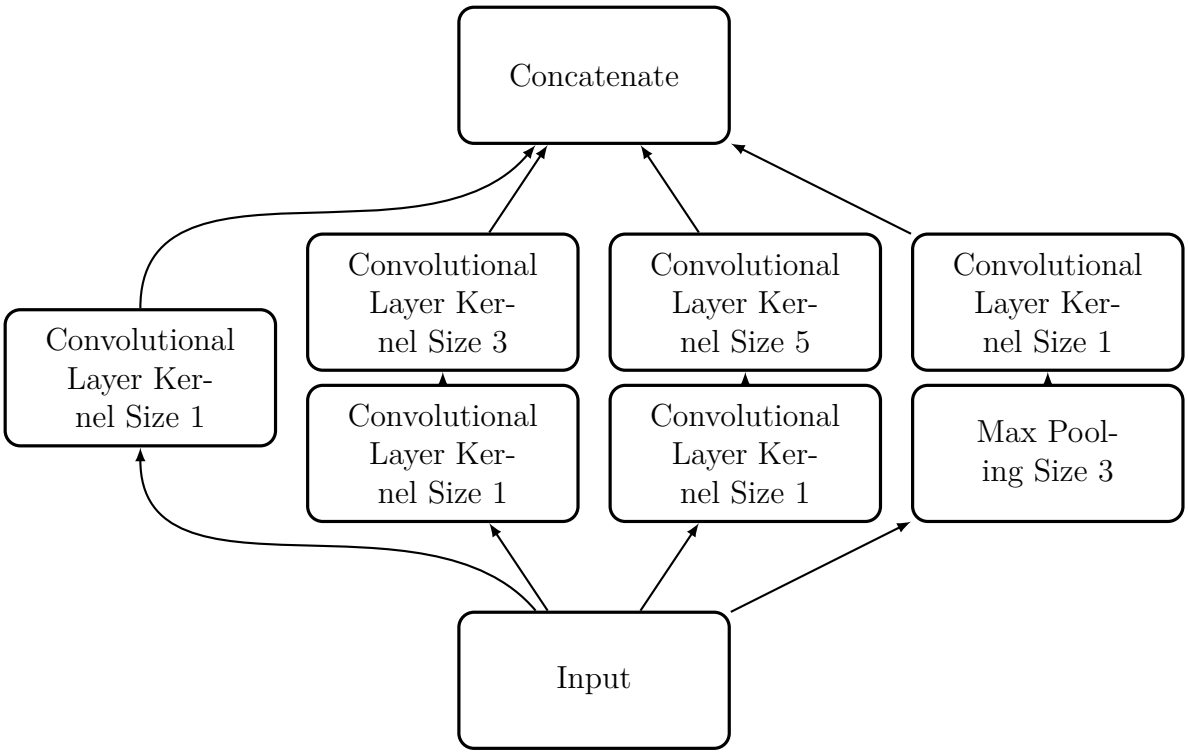



Figure 4.8: Inception Module (own figure)

Based on this, inception modules can be generalized. A generalized inception module is comprised of several paths. The paths branch from the input of the inception module and merge to the output of the inception module.[112]

The best-performing inception network found in the course of the literature review is the 101 version of S. Xie et al. 2017's ResNeXt, called ResNeXt-101. This version is based on the 101-layer version of ResNet. The input of ResNeXt-101 is a 224-by-224-pixel, RGB image. The output of ResNeXt-101 comprises the probabilities of the $c$ target classes.[113]

ResNeXt-101 is comprised of convolutional layers followed by 1 dense layer. The convolutional layers have a stride of 1, unless specified otherwise. The convolutional layers use batch normalization and the ReLU activation function. Batch normalization

[111]Szegedy, W. Liu, et al. 2015.
[112]Szegedy, Vanhoucke, et al. 2016; Szegedy, Ioffe, et al. 2017; S. Xie et al. 2017.
[113]S. Xie et al. 2017.

is applied after every convolution before the activation function. The dense layer has $c$ neurons and uses the softmax activation function. The first convolutional layer is followed by max pooling, has a kernel size of 7, and a stride of 2. Max pooling is applied with a pooling size of 3 and a pooling stride of 2.[114]

The remaining covolutional layers are arranged in inception modules with shortcut connections. An inception module consists of 32 paths. A path consists of three convolutional layers. The first and third layer have a kernel size of 1. The second layer has a kernel size of 3. The first and third layer are used to reduce, then increase (restore) dimensions. This way, the second layer becomes a bottleneck with smaller input/output dimensions. On that account, computation costs are reduced. The shortcut connection skips the inception module. The shortcut connection uses identity mapping. The output of the identity mapping is added element-wise to the output of the skipped inception module. The result is transformed by the ReLU activation function.[115] ResNeXt-101 consists of 4 types of stacked inception modules . The types differ only in the number of kernels of their layers. The configurations of each inception module are outlined in Table 4.3.[116]

The first layer of the first block of a type has a stride of 2 for all paths. This way, the feature map size is halved while the number of filters is doubled. Hence, the time complexity per inception module is preserved. The last layer of the last block is followed by global average pooling.[117]

The whole configuration of ResNeXt-101 is outlined in Table 4.3.[118]

Table 4.3: ResNeXt-101 Configuration. Note that each $k \times k$ conv $K$ denotes a convolutional layer with $K$ kernels of size $k$. An inception module is denoted as an array of layers and $P = p$ with $p$ being the number of paths.

| **Layer/Module** | **Configuration** |
|---|---|
| Input Layer | $7 \times 7, 64$ with stride of 2, and $3 \times 3$ max pooling with stride of 2 |
| Inception Modul $1 - 3$ | $\begin{bmatrix} 1 \times 1 \text{ conv } 4 \\ 3 \times 3 \text{ conv } 4 \quad P = 32 \\ 1 \times 1 \text{ conv } 256 \end{bmatrix} \times 3$ |
| Inception Modul $4 - 7$ | $\begin{bmatrix} 1 \times 1 \text{ conv } 4 \\ 3 \times 3 \text{ conv } 4 \quad P = 32 \\ 1 \times 1 \text{ conv } 512 \end{bmatrix} \times 4$ |

[114] S. Xie et al. 2017.
[115] S. Xie et al. 2017.
[116] S. Xie et al. 2017.
[117] S. Xie et al. 2017.
[118] S. Xie et al. 2017.

Table 4.3: ResNeXt-101 Configuration. Note that each $k \times k$ conv $K$ denotes a convolutional layer with $K$ kernels of size $k$. An inception module is denoted as an array of layers and $P = p$ with $p$ being the number of paths.

| **Layer/Module** | **Configuration** |
|---|---|
| Inception Modul $8 - 30$ | $\begin{bmatrix} 1 \times 1 \text{ conv } 4 \\ 3 \times 3 \text{ conv } 4 \quad P = 32 \\ 1 \times 1 \text{ conv } 1024 \end{bmatrix} \times 23$ |
| Inception Modul $31 - 33$ | $\begin{bmatrix} 1 \times 1 \text{ conv } 4 \\ 3 \times 3 \text{ conv } 4 \quad P = 32 \\ 1 \times 1 \text{ conv } 2048 \end{bmatrix} \times 3$ |
| Output Layer | global average pooling, dense with $c$ neurons |

## 4.4 Dense Convolutional Neural Network

Dense convolutional neural networks address a problem of deep CNNs.[119]

> As information about the input or gradient passes through many layers, it can vanish and "washout" by the time it reaches the end (or beginning) of the network. (G. Huang et al. 2017)

Dense convolutional neural networks address this problem by connecting all layers with each other, see Figure 4.9. The input of a layer is the concatenation of the original input and all outputs of previous layers. This way, information flow between layers in the network is encouraged.[120]

[119] G. Huang et al. 2017.
[120] G. Huang et al. 2017.

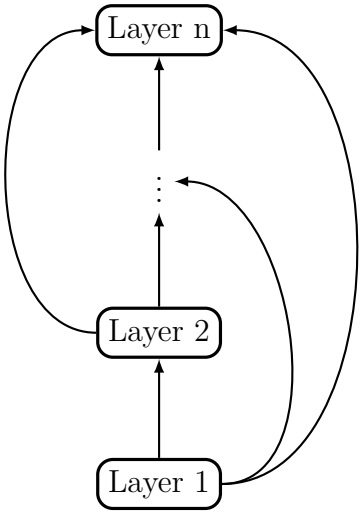



Figure 4.9: Dense Convolutional Neural Network (own figure)

An additional, counter-intuitive effect of dense convolutional neural networks is that they require fewer parameters. Traditional neural networks pass information from layer to layer. Consequently, layers need to learn what information needs to be added and preserved. Dense convolutional neural networks explicitly distinguish between added and preserved information. Features are reused. Thus, they do not need to learn which information to preserve. Therefore, they need fewer parameters.[121]

The best-performing dense convolutional neural network found in the course of the literature review is the 264 layer version of G. Huang et al. 2017's DenseNet, called DenseNet-264. The input of DenseNet-264 is a 224-by-224-pixel, RGB image. The output of DenseNet-264 comprises the probabilities of the $c$ target classes. DenseNet-264 is comprised of convolutional layers followed by 1 dense layer.[122]

The convolutional layers have a stride of 1 and a padding preserving spatial dimensions. The first convolutional layer has 64 kernels of size 7 and a stride of 2. It is followed by batch normalization, ReLU activation function, and max pooling. Max pooling is applied with a pooling size of 3, a pooling stride of 2, and a padding preserving spatial dimensions.[123] The remaining convolutional layers are arranged in 4 dense blocks and 3 transition layers. A transition layer is applied between each dense block. After the last dense block batch normalization, ReLU activation function, global average pooling, and a dense layer are applied. The dense layer has $c$ neurons and uses the softmax activation function.[124]

[121]G. Huang et al. 2017.
[122]G. Huang et al. 2017.
[123]S. Xie et al. 2017.
[124]G. Huang et al. 2017.

A dense block consists of $L$ convolutional blocks. All convolutional blocks are connected with each other. In consequence, the input of the $l$th convolutional block is the concatenation of the input of the dense block and the outputs of all previous layers. The output of the dense block is the concatenation of the input of the dense block and the outputs of all convolutional blocks of the dense block. Therefore, inside the dense block, the spatial dimensions of all feature maps are the same. A convolutional block consists of 2 convolutional layers. The first layer has 128 kernels of size 1. The second layer has 32 kernels of size 3. Batch normalization and ReLU activation function are applied before each convolution. The configurations of each dense block are outlined in Table 4.4.[125]

A transition layer reduces the number and spatial dimensions of the feature maps. A transition layer consists of the sequence batch normalization, ReLU activation function, convolution, and average pooling. The convolution is used to reduce the number of feature maps by factor 0.5. Hence, the convolution has $0.5 \cdot d$ kernels of size 1 with $d$ being the initial number of feature maps. The average pooling is used to reduce the spatial dimensions of the feature maps. The pooling is of size 2 and has a stride of 2.[126]

The whole configuration of DenseNet-264 is outlined in Table 4.4.[127]

Table 4.4: DenseNet-264 Configuration. Note that each $k \times k$ conv $K$ denotes the sequence batch normalization, ReLU, and convolution with $K$ kernels of size $k$, except for the first conv, which denotes the sequence convolution, batch normalization ReLU.

| **Layer/Block** | **Configuration** |
|---|---|
| Input Layer | $7 \times 7$ conv 64, stride 2, and $3 \times 3$ max pooling, stride 2 |
| Dense Block 1 | $\begin{bmatrix} 1 \times 1 \text{ conv } 128 \\ 3 \times 3 \text{ conv } 32 \end{bmatrix} \times 6$ |
| Transition Layer 1 | $1 \times 1$ conv $0.5d$, stride 2, and $2 \times 2$ average pooling, stride 2 |
| Dense Block 2 | $\begin{bmatrix} 1 \times 1 \text{ conv } 128 \\ 3 \times 3 \text{ conv } 32 \end{bmatrix} \times 12$ |
| Transition Layer 2 | $1 \times 1$ conv $0.5d$, stride 2, and $2 \times 2$ average pooling, stride 2 |
| Dense Block 3 | $\begin{bmatrix} 1 \times 1 \text{ conv } 128 \\ 3 \times 3 \text{ conv } 32 \end{bmatrix} \times 64$ |
| Transition Layer 3 | $1 \times 1$ conv $0.5d$, stride 2, and $2 \times 2$ average pooling, stride 2 |

[125] G. Huang et al. 2017.
[126] G. Huang et al. 2017.
[127] G. Huang et al. 2017.

Table 4.4: DenseNet-264 Configuration. Note that each $k \times k$ conv $K$ denotes the sequence batch normalization, ReLU, and convolution with $K$ kernels of size $k$, except for the first conv, which denotes the sequence convolution, batch normalization ReLU.

| **Layer/Block** | **Configuration** |
|---|---|
| Dense Block 4 | $\begin{bmatrix} 1 \times 1 \text{ conv } 128 \\ 3 \times 3 \text{ conv } 32 \end{bmatrix} \times 48$ |
| Output Layer | batch normalization, ReLU, global average pooling, and dense with $c$ neurons |

## 4.5 Capsule Network

Capsule networks are neural networks utilizing capsules. A capsule is a group of neurons. A capsule detects the properties of a specific object, e.g., rectangle. These properties are called instantiation parameters, e.g., position. If capsules are stacked, a higher-level capsule detects objects composed of objects detected by lower-level capsules in a specific relation. Capsules detecting an object are called active. For a given input, a capsule is active if the corresponding object is present. Hence, the activated capsules of a network form a tree carved out of the network. The mechanism deciding which capsule is activated is called routing mechanism. Capsules and routing mechanism can be implemented in various ways. If an object represented by a capsule is present, the capsule is called active.[128]

The best-performing capsule network found in the course of the literature review is Sabour, Frosst, and G. E. Hinton 2017's CapsNet.[129]

Sabour, Frosst, and G. E. Hinton 2017 implement capsules by representing the instantiation parameters of an object as a vector. The existence of an object is represented by a probability. The probability is represented by the length of the output vector of the capsule. Therefore, the length must be in the range of $[0; 1]$.[130] Capsules in the first layer of capsules are called primary capsules. Primary capsules are implemented as a convolutional layer using the *squash* activation function defined by Equation (4.2).[131]

$$squash(x) = \frac{||x||^2}{1 + ||x||^2} \frac{x}{||x||} \tag{4.2}$$

[128]Sabour, Frosst, and G. E. Hinton 2017.
[129]Sabour, Frosst, and G. E. Hinton 2017.
[130]Sabour, Frosst, and G. E. Hinton 2017.
[131]Sabour, Frosst, and G. E. Hinton 2017.

Given the log probability $b_{ij}$ of capsule $i$ in layer $l$ coupling with capsule $j$ in layer $l+1$ and the input $u_i$ of capsule $j$, the output vector $v_j$ of a capsule $j$ is computed as described by Equation (4.3). The coupling coefficient $c_{ij}$ determines whether or not capsules $i$ and $j$ are coupled. For example, $c_{ij} = 0$ means $i$ and $j$ are not coupled. Coupling is implemented by weighting the inputs by the coupling coefficients, see $s_j$ in Equation (4.3). The *squash* activation function scales the magnitude of a vector in the range of $[0; 1]$ while leaving its orientation unchanged.[132]

$$\begin{array}{lcl} v_j & = & squash(s_j) \\ s_j & = & \sum_i c_{ij}\hat{u}_{ji} \\ \hat{u}_{ji} & = & W_{ij}u_i \\ c_{ij} & = & softmax(b_{ij}) \end{array} \tag{4.3}$$

Sabour, Frosst, and G. E. Hinton 2017 implement a routing mechanism called dynamic routing mechanism. Active capsules of a layer predict the output (instantiation parameters) of a capsule in the next layer. When multiple predictions agree, the capsule becomes active. This mechanism is implemented as follows: Initially, the output of all capsules is routed to all capsules in the succeeding layer. The prediction is computed by multiplying the output of a capsule by a weight matrix. The level of agreement is measured by the scalar product of prediction and actual output. A huge scalar product increases the coupling of the corresponding capsules while decreasing the other couplings. This is repeated $r$ times. Dynamic routing between capsules is formalized in Algorithm 1.[133]

**Algorithm 1** Dynamic Routing Between Capsules

1: **procedure** ROUTING($\hat{u}_{ji}, r, l$)
2: for all capsules $i$ in layer $l$ and capsules $j$ in layer $l+1$: $b_{ij} \leftarrow 0$
3: **for** $r$ iterations **do**
4: for all capsules $i$ in layer $l$: $c_i \leftarrow softmax(b_i)$
5: for all capsules $j$ in layer $l+1$: $s_j \leftarrow \sum_i c_{ij}\hat{u}_{ji}$
6: for all capsules $j$ in layer $l+1$: $v_j \leftarrow squash(s_j)$
7: for all capsules $i$ in layer $l$ and capsules $j$ in layer $l+1$: $b_{ij} \leftarrow b_{ij} + \hat{u}_{ji}v_j$
8: **end for**
9: **return** $v_j$
10: **end procedure**

The input of CapsNet is a 28-by-28-pixel, grayscale image. The output of CapsNet comprises the probabilities of the $c$ target classes. CapsNet is comprised of a convo-

[132] Sabour, Frosst, and G. E. Hinton 2017.
[133] Sabour, Frosst, and G. E. Hinton 2017.

lutional layer followed by 2 capsule layers.[134] The convolutional layer has 256 kernels of size 9, a stride of 1, and the ReLU activation function. The primary capsules have 32 kernels of size 9 and a stride of 2. Each primary capsule is composed of 8 kernels. The final capsule layer is comprised of $c$ capsules. Each capsule is composed of 16 neurons. Routing is only applied between the capsule layers.[135]

## 4.6 Depthwise Separable Convolutional Neural Network

Depthwise separable convolutional neural networks are CNNs using depthwise separable convolution. Depthwise separable convolution factorizes a standard convolution into a depthwise convolution and a pointwise convolution. The factorization requires less parameters than the standard convolution. Therefore, it is more computationally effective. The depthwise convolution is a standard convolution which applies a single kernel per input channel. As a result, the output consists of one feature map per input channel. The pointwise convolution is a standard convolution of kernels size 1. Given an input $X$ consisting of $d$ channels and a convolution $conv_{k\times k}$ of kernel size $k$, the depthwise separable convolution $sepconv_{k\times k}$ corresponding to $conv_{k\times k}$ is defined by Equation (4.4).[136] Depthwise convolution is illustrated in Figure 4.10. Pointwise convolutions is illustrated in Figure 4.11

$$\begin{array}{lcl} sepconv(X) & = & pointwiseconv(depthwiseconv(X)) \\ depthwiseconv(X) & = & concat(depthwiseconv_0, \ldots, depthwiseconv_{d-1}) \\ depthwiseconv_i & = & conv_{k\times k}(X_i) \\ pointwiseconv(X) & = & conv_{1\times 1}(X) \end{array} \tag{4.4}$$

[134] Note that for training CapsNet uses an auxiliary neural network on top. Such auxiliary training techniques are excluded from the scope of this paper, see Section 1.3. Thus, they are not discussed further.

[135] Sabour, Frosst, and G. E. Hinton 2017.

[136] Yunhui Guo et al. 2019; Chollet 2017.

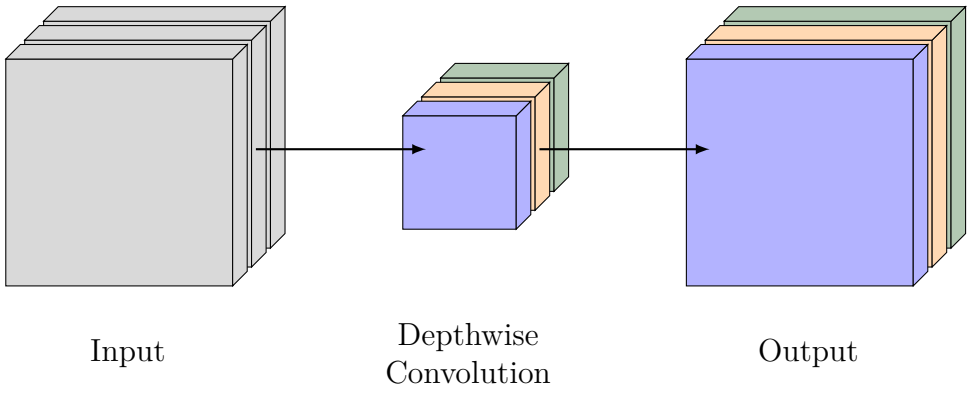



Figure 4.10: Depthwise Convolution Illustration (own figure)

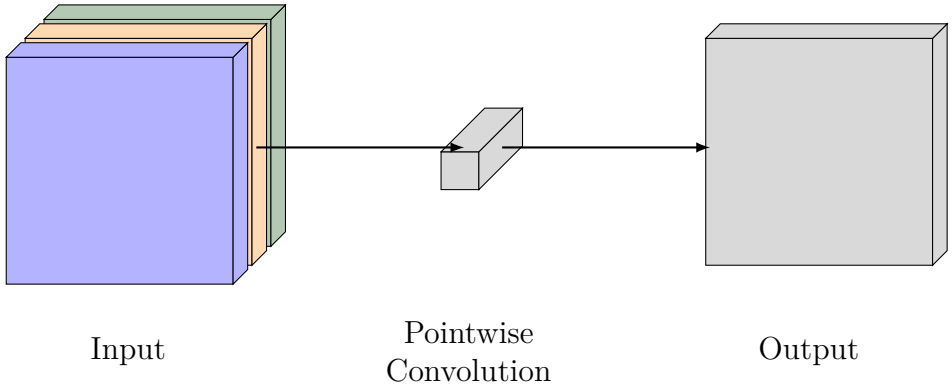



Figure 4.11: Pointwise Convolution Illustration (own figure)

The best-performing depthwise separable convolutional neural network found in the course of the literature is Tan and Le V 2019's EfficientNet-B7. EfficientNet-B7 is a scaled version of Tan and Le V 2019's EfficientNet-B0.

Scaling a neural network refers to increasing its depth, width, and image size. Depth $d$ refers to the number of layers of a neural network. Width $w$ refers to the number of input channels of a layer. Image size $r$ refers to the resolution of the input image. Scaling a neural networks is used to increase its accuracy.[137]

> Intuitively, for higher resolution images, we should increase network depth, such that the larger receptive fields can help capture similar features that include more pixels in bigger images. Correspondingly, we should also increase network width when resolution is higher, in order to capture more fine-grained patterns with more pixels in high resolution images. (Tan and Le V 2019)

[137] Tan and Le V 2019.

Thus, balancing depth, width, and image size is necessary. Tan and Le V 2019 propose a compound scaling method to balance scaling of depth, width, and image size. The compound scaling method uses a compound coefficient $\phi$ to scale $d$, $w$, and $r$ as defined by Equation (4.5). Tan and Le V 2019 find $\alpha = 1.2, \beta = 1.1, \gamma = 1.15$.[138]

$$\begin{aligned} d &= \alpha^{\phi} \\ w &= \beta^{\phi} \\ r &= \gamma^{\phi} \end{aligned} \tag{4.5}$$

EfficientNet-B0 is comprised of a stem, followed by mobile inverted bottleneck blocks, and a top. The input of EfficientNet-B0 is a 224-by-224-pixel, RGB image. The output of EfficientNet-B0 comprises the probabilities of the $c$ target classes.[139]

The stem consists of a convolutional layer and batch normalization. The convolutional layer has 32 kernels of size 3, a padding halving the spatial dimensions, and the swish activation function.[140]

The top consists of a convolutional layer, global average pooling, dropout, and a dense layer. The convolutional layer has 1,280 kernels of size 1, a padding preserving the spatial dimensions, and the swish activation function. Batch normalization is applied before the swish activation function. The dropout rate is 0.2. The dense layer has $c$ neurons and uses the softmax activation function.[141]

A mobile inverted bottleneck block consists of an expansion phase, depthwise convolution, a squeeze-and-excitation phase, an output phase, and a shortcut connection. EfficientNet-B0 is comprised of 7 types of mobile inverted bottleneck blocks. The types are defined by the following hyperparameters:[142]

- Kernel size of the depthwise convolution $k$
- Number of stacked mobile inverted bottleneck blocks of a type *repeats*
- Number of input feature maps of the first mobile inverted bottleneck block of a type $K_{in}$, $K_{in} = K_{out}$ for any block except the first block of a type
- Number of output feature maps of a mobile inverted bottleneck block $K_{out}$
- Ratio by which $K_{in}$ is increased in the expansion phase $ratio_{expad}$
- Stride of the depthwise convolution $stride_{depthconv}$
- Ratio by which $K_{in}$ is squeezed in the squeeze-and-excitation phase $ratio_{squeeze}$

[138] Tan and Le V 2019.
[139] Tan and Le V 2019.
[140] Tan and Le V 2019.
[141] Tan and Le V 2019.
[142] Tan and Le V 2019.

The expansion phase expands the number of input feature maps $K_{in}$ by $ratio_{expad}$ to $K = K_{in} \cdot ratio_{expand}$. The expansion is implemented using a convolutional layer with $K$ kernels of size 1. The convolutional layer uses a padding preserving the spatial dimensions, batch normalization, and the swish activation function. Batch normalization is applied before the swish activation function. The depthwise convolution has a kernel size of $k$, a stride of $stride_{depthconv}$, a padding preserving spatial dimensions, batch normalization, and the swish activation function. Batch normalization is applied before the swish activation function. The squeeze-and-excitation phase consists of global average pooling and two convolutional layers. The first convolutional layer reduces the number of filters to $K_{in} \cdot ratio_{squeeze}$. The second convolutional layer expands (restores) the number of filters to $K$. Expansion and reduction are implemented by convolutions with kernel size of 1. The output phase consists of a convolutional layer, batch normalization, and dropout. The convolutional layer has $K_{out}$ kernels of size 1 and a padding to preserve spatial dimensions. The dropout rate increases with network depth. The dropout rate for the $n$th block is $0.2 \cdot \frac{n}{N}$ with $N$ being the total number of blocks. The shortcut connection skips the mobile inverted bottleneck block. The shortcut connection uses identity mapping. The output of the identity mapping is added element-wise to the output of the skipped mobile inverted bottleneck block.

The whole configuration of EfficientNet-B0 is outlined in Table 4.5.[143]

Table 4.5: EfficientNet-B0 Configuration

| **Layer/Block** | **Configuration** |
|---|---|
| Stem | |
| Mobile Inverted Bottleneck Block Type 1 | $k = 3, repeats = 1, K_{in} = 32, K_{out} = 16, ratio_{expad} = 1, stride_{depthconv} = 1, ratio_{squeeze} = 0.25$ |
| Mobile Inverted Bottleneck Block Type 2 | $k = 3, repeats = 2, K_{in} = 16, K_{out} = 24, ratio_{expad} = 6, stride_{depthconv} = 2, ratio_{squeeze} = 0.25$ |
| Mobile Inverted Bottleneck Block Type 3 | $k = 5, repeats = 2, K_{in} = 24, K_{out} = 40, ratio_{expad} = 6, stride_{depthconv} = 2, ratio_{squeeze} = 0.25$ |
| Mobile Inverted Bottleneck Block Type 4 | $k = 3, repeats = 3, K_{in} = 40, K_{out} = 80, ratio_{expad} = 6, stride_{depthconv} = 2, ratio_{squeeze} = 0.25$ |

[143] Tan and Le V 2019.

Table 4.5: EfficientNet-B0 Configuration

| **Layer/Block** | **Configuration** |
|---|---|
| Mobile Inverted Bottleneck Block Type 5 | $k = 5, repeats = 3, K_{in} = 80, K_{out} = 112, ratio_{expad} = 6, stride_{depthconv} = 1, ratio_{squeeze} = 0.25$ |
| Mobile Inverted Bottleneck Block Type 6 | $k = 5, repeats = 4, K_{in} = 112, K_{out} = 192, ratio_{expad} = 6, stride_{depthconv} = 2, ratio_{squeeze} = 0.25$ |
| Mobile Inverted Bottleneck Block Type 7 | $k = 3, repeats = 1, K_{in} = 192, K_{out} = 320, ratio_{expad} = 6, stride_{depthconv} = 1, ratio_{squeeze} = 0.25$ |

EfficientNet-B7 is EfficientNet-B0 scaled by $\phi = 7$. Depth is scaled by increasing *repeats* by $\alpha^{\phi}$. Width is scaled by increasing the number of kernels for every layer in the whole network by $\beta^{\phi}$. Input size is scaled by increasing the resolution of the input image by $\gamma^{\phi}$.[144] Furthermore, the dropout rate is increased to 0.5.

[144]Note that *repeats*, number of kernels, and resolution must be natural numbers. Hence, rounding might be necessary.

# 5 Results

This paper seeks to determine the best-performing neural network for tool image classification. The best-performing neural network was determined in the course of an experiment. The experiment is described in Section 3.4. The experiment was conducted with selected neural networks. The performances of the selected neural networks were measured in accuracy. The accuracy for each neural network is reported in Table 5.1. Among these neural networks, DenseNet-264 performs best for the TIC Dataset. On that account, DenseNet-264 is determined as the best-performing neural network for tool image classification. ResNet-152, ResNeXt-101, and DenseNet-264 perform rather similar. As a result, in general, several neural networks are suitable for tool image classification.

As stated in Section 3.4, the neural network from the epoch with the highest validation performance is evaluated. On that account, those epochs are reported for reasons of transparency as well. The number of epochs is reported in Table 5.2.

As stated in Section 3.4, EfficientNet-B7 was trained for the remaining time after the other neural networks were trained. EfficientNet-B7 was trained for 100 hours in total. 100 hours were sufficient to train EfficientNet-B7 for 20 epochs. 20 epochs were not sufficient to train EfficientNet-B7 until convergence.

Table 5.1: Experiment Results

| **Neural Network** | **Accuracy in %** |
|---|---|
| VGG-19 | 72.40 |
| ResNet-152 | 92.89 |
| ResNeXt-101 | 94.66 |
| **DenseNet-264** | **97.45** |
| EfficientNet-B7 | 16.67 |

Table 5.2: Number of Epochs

| **Neural Network** | **Number of Epochs** |
|---|---|
| VGG-19 | 56 |
| ResNet-152 | 90 |
| ResNext-101 | 43 |

Table 5.2: Number of Epochs

| **Neural Network** | **Number of Epochs** |
|---|---|
| DenseNet-264 | 81 |
| EfficientNet-B7 | 20 |

# 6 Discussion

Augmented reality solutions for field workers are an emerging market.[145] An augmented reality solution for field workers requires software perceiving the environment of field workers. A sub-task of that perception is to classify tools of different classes. This paper determined the best-performing neural network for tool image classification in the course of an experiment. The experiment trains and evaluates state-of-the-art neural networks for image classification on the TIC Dataset. The state-of-the-art neural networks for image classification are determined in the course of a literature review conducted by this paper. The best-performing neural network found in the course of the experiment is DenseNet-264. However, DenseNet-264 is only the best-performing neural network for the TIC Dataset and under the constraints of this paper. This paper found that, in general, several neural networks are suitable for tool image classification. This is in accordance with the state of the art of image classification, as image classification leaderboards show. Image classification leaderboards show that different neural networks perform rather similarly for different image classification tasks.[146] This paper found ResNet-152, ResNeXt-101, and DenseNet-264 to be suitable for tool image classification. This was to be expected since, ResNet-152, ResNeXt-101, and DenseNet-264 are suitable for image classification in general.[147]

ResNet-152, ResNeXt-101, and DenseNet-264 performed rather similar. Despite differing in structure, each of these neural networks uses convolutional layers and connections skipping layers.[148] This might be the cause of the similar performance. In comparison to these neural networks, VGG-19 performs less well. VGG-19 does not use connections skipping layers.[149] These connections encourage information flow in neural networks. Based on this, these connections ease optimization.[150] Not using these connections might be the cause of the lower performance of VGG-19. EfficientNet-B7 achieved an accuracy of 16.67%. The TIC Dataset is comprised of six balanced classes. In consequence, the probability of guessing a class correctly is 16.67%. This

[145] Ernst & Young 2019a; Ernst & Young 2019b; Detzel et al. 2018; Shook and Knickrehm 2017; Guy et al. 2019.

[146] Deng et al. 2009; Krizhevsky 2012; LeCun, Cortes, and Burges 2010; Netzer et al. 2011; Coates, Ng, and H. Lee 2011; Z. Liu et al. 2016; H. Xiao, Rasul, and Vollgraf 2017; Darlow et al. 2018; Nilsback and A. Zisserman 2008; Bossard, Guillaumin, and Van Gool 2014; van Horn et al. 2018; Krause et al. 2013; Cohen et al. 2017; Clanuwat et al. 2018; Wah et al. 2011; Sabour, Frosst, and G. E. Hinton 2017; Combalia et al. 2019; Codella et al. 2018; Tschandl, Rosendahl, and Kittler 2018.

[147] S. Xie et al. 2017; K. He et al. 2016; G. Huang et al. 2017.

[148] S. Xie et al. 2017; K. He et al. 2016; G. Huang et al. 2017.

[149] Simonyan and Andrew Zisserman 2014.

[150] K. He et al. 2016; G. Huang et al. 2017.

means, EfficientNet-B7 did not learn to classify tool images, instead EfficientNet-B7 just guessed.

The experiment was conducted with state-of-the-art neural networks for image classification. On that account, the results of the experiment for each neural network are individually discussed based on the results reported in the original paper proposing the neural network. The results of the experiment for each neural network are individually discussed in the following sections.

## 6.1 VGG-19

According to the original paper, VGG-19 achieved an accuracy of 76.3% for the ImageNet dataset.[151] In the experiment conducted by this paper, VGG-19 achieved an accuracy of 72.40%.

The original paper applied data augmentation in the form of sampling input crops from multi-scale training images, random horizontal flipping, and random color shifting.[152] Data augmentation creates additional training data. Additional training data can improve performance.[153] Data augmentation is excluded from the scope of this paper, see Section 1.3. Therefore, data augmentation was not used in the course of the experiment.

Furthermore, Simonyan and Andrew Zisserman 2014 used hardware providing enough GPU RAM to train VGG-19 using a batch size of 256. The used batch size is eight times larger than the batch size of 32 used in the course of the experiment. A larger batch size decreases fluctuation of the loss function supporting convergence to the optimum.[154]

Furthermore, Simonyan and Andrew Zisserman 2014 used weight decay. Weight decay improves training by penalizing large weights.[155] Weight decay is excluded from the scope of this paper, see Section 1.3. Thus, weight decay was not used in the course of the experiment.

Finally, the ImageNet dataset and the TIC Dataset differ. The ImageNet dataset provides 1.28 million training images. This is over 100 times more than the TIC Dataset

[151] Simonyan and Andrew Zisserman 2014.
[152] Simonyan and Andrew Zisserman 2014.
[153] El-Amir and Hamdy 2020.
[154] Ruder 2016.
[155] El-Amir and Hamdy 2020.

with 12,240 training images.[156] More training data can improve performance.[157] Images, classes, and number of classes differ for both tasks. The ImageNet dataset has 1,000 classes while the TIC Dataset has six classes. Hence, classification for both datasets might be differently complex.

As a result, data augmentation, hardware providing more GPU RAM, weight decay, and different tasks might be the cause of the difference in accuracy. Note that classifying 1,000 classes is probably more complex than classifying six classes. Accordingly, data augmentation, hardware providing more GPU RAM, weight decay, and more training data might compensate the more complex task.

## 6.2 ResNet-152

According to the original paper, ResNet-152 achieved an accuracy of 78.57% for the ImageNet dataset.[158] In the experiment conducted by this paper, ResNet-152 achieved an accuracy of 92.89%.

The original paper applied input normalization and data augmentation. Input normalization was applied in the form of subtracting the per-pixel mean. Data augmentation was applied in the form of sampling input crops from multi-scale training images, random horizontal flipping, and random color shifting.[159] Data augmentation creates additional training data. Additional training data and input normalization can improve performance.[160] Input normalization and data augmentation are excluded from the scope of this paper, see Section 1.3. Therefore, input normalization and data augmentation were not used in the course of the experiment.

Furthermore, K. He et al. 2016 used hardware providing enough GPU RAM to train ResNet-152 using a batch size of 256. The used batch size is eight times larger than the batch size of 32 used in the course of the experiment. A larger batch size decreases fluctuation of the loss function supporting convergence to the optimum.[161]

Furthermore, K. He et al. 2016 used weight decay. Weight decay improves training by penalizing large weights.[162] Weight decay is excluded from the scope of this paper, see Section 1.3. Thus, weight decay was not used in the course of the experiment.

[156] K. He et al. 2016.
[157] El-Amir and Hamdy 2020.
[158] K. He et al. 2016.
[159] K. He et al. 2016.
[160] El-Amir and Hamdy 2020.
[161] Ruder 2016.
[162] El-Amir and Hamdy 2020.

Finally, the ImageNet dataset and the TIC Dataset differ. The ImageNet dataset provides 1.28 million training images. This is over 100 times more than the TIC Dataset with 12,240 training images.[163] More training data can improve performance.[164] Images, classes, and number of classes differ for both tasks. The ImageNet dataset has 1,000 classes while the TIC Dataset has six classes. Hence, classification for both datasets might be differently complex.

The accuracy achieved in the original paper is lower despite the use of data augmentation, hardware providing more GPU RAM, weight decay, and more training data. As a result, the cause of the lower accuracy might be that classification for both datasets is differently complex.

## 6.3 ResNeXt-101

According to the original paper, ResNeXt-101 achieved an accuracy of 79.6% for the ImageNet dataset with 1,000 classes and an accuracy of 59.9% for the ImageNet dataset with 5,000 classes.[165] In the experiment conducted by this paper, ResNeXt-101 achieved an accuracy of 94.66%.

The original paper applied data augmentation in the form of sampling input crops from multi-scale training images.[166] Data augmentation creates additional training data. Additional training data can improve performance.[167] Data augmentation is excluded from the scope of this paper, see Section 1.3. Therefore, data augmentation was not used in the course of the experiment.

Furthermore, S. Xie et al. 2017 used hardware providing enough GPU RAM to train ResNeXt-101 using a batch size of 256. The used batch size is 16 times larger than the batch size of 16 used in the course of the experiment. A larger batch size decreases fluctuation of the loss function supporting convergence to the optimum.[168]

Furthermore, S. Xie et al. 2017 used weight decay. Weight decay improves training by penalizing large weights.[169] Weight decay is excluded from the scope of this paper, see Section 1.3. Thus, weight decay was not used in the course of the experiment.

Finally, the ImageNet dataset and the TIC Dataset differ. The ImageNet dataset provides 1.28 million training images for 1,000 classes and 6.8 million training images

[163] K. He et al. 2016.
[164] El-Amir and Hamdy 2020.
[165] S. Xie et al. 2017.
[166] S. Xie et al. 2017.
[167] El-Amir and Hamdy 2020.
[168] Ruder 2016.
[169] El-Amir and Hamdy 2020.

for 5,000 classes. This is significantly more than the TIC Dataset with 12,240 training images.[170] More training data can improve performance.[171] Images, classes, and number of classes differ for the tasks. S. Xie et al. 2017 used the ImageNet dataset with 1,000 and 5,000 classes. The more classes the lower was the accuracy. Hence, classification for the datasets might be differently complex.

The accuracies achieved in the original paper are lower despite the use of input normalization, data augmentation, hardware providing more GPU RAM, weight decay, and more training data. As a result, the cause of the lower accuracies might be that classification for the datasets is differently complex.

## 6.4 DenseNet-264

According to the original paper, DenseNet-264 achieved an accuracy of 77.85% for the ImageNet dataset.[172] In the experiment conducted by this paper, DenseNet-264 achieved an accuracy of 97.45%.

The original paper applied input normalization and data augmentation. Input normalization was applied in the form of subtracting the per-pixel mean. Data augmentation was applied in the form of sampling input crops from multi-scale training images, random horizontal flipping, and random color shifting.[173] Data augmentation creates additional training data. Additional training data and input normalization can improve performance.[174] Input normalization and data augmentation are excluded from the scope of this paper, see Section 1.3. Therefore, input normalization and data augmentation were not used in the course of the experiment.

Furthermore, G. Huang et al. 2017 used hardware providing enough GPU RAM to train DenseNet-264 using a batch size of 256. The used batch size is eight times larger than the batch size of 32 used in the course of the experiment. A larger batch size decreases fluctuation of the loss function supporting convergence to the optimum.[175]

Furthermore, G. Huang et al. 2017 used weight decay. Weight decay improves training by penalizing large weights.[176] Weight decay is excluded from the scope of this paper, see Section 1.3. Thus, weight decay was not used in the course of the experiment.

[170] K. He et al. 2016.
[171] El-Amir and Hamdy 2020.
[172] G. Huang et al. 2017.
[173] G. Huang et al. 2017.
[174] El-Amir and Hamdy 2020.
[175] Ruder 2016.
[176] El-Amir and Hamdy 2020.

Finally, the ImageNet dataset and the TIC Dataset differ. The ImageNet dataset provides 1.28 million training images. This is over 100 times more than the TIC Dataset with 12,240 training images.[177] More training data can improve performance.[178] Images, classes, and number of classes differ for both tasks. The ImageNet dataset has 1,000 classes while the TIC Dataset has six classes. Hence, classification for both datasets might be differently complex.

The accuracy achieved in the original paper is lower despite the use of input normalization, data augmentation, hardware providing more GPU RAM, weight decay, and more training data. As a result, the cause of the lower accuracy might be that classification for both datasets is differently complex.

## 6.5 EfficientNet-B7

According to the original paper, EfficientNet-B7 achieved an accuracy of 84.4% on the ImageNet dataset.[179] In the experiment conducted by this paper, EfficientNet-B7 did not learn to classify tool images. In the course of the experiment, EfficientNet-B7 was trained using a batch size of one and a drop out rate of over 50% for some layers. A batch size of one causes the loss function to fluctuate heavily. This impairs convergence to the optimum. Dropout is a regularization technique. The impaired convergence and high regularization might have prevented EfficientNet-B7 from learning properly.

Furthermore, in the original paper, EfficientNet-B7 was trained for more training time, on more training data, and used data augmentation.[180] Data augmentation creates additional training data. Additional training time and training data can improve performance.[181] However, in the course of the experiment EfficientNet-B7 did not even start to learn to classify tool images. For this reason, this paper regards it unlikely that less training time and less training data is the reason why EfficientNet-B7 did not learn to classify tool images.

## 6.6 Placement in the State of the Art

This paper builds upon the state of the art of image classification to determine the best-performing neural network for tool image classification. The neural network is

[177] G. Huang et al. 2017.
[178] El-Amir and Hamdy 2020.
[179] Tan and Le V 2019.
[180] Tan and Le V 2019.
[181] El-Amir and Hamdy 2020.

determined in the course of an experiment. The methodology of this paper, i.e., metric determination, experiment design, and literature review follows, common practice. The metric for the experiment is chosen based on the metrics used by image classification leaderboards, see Section 3.1. The experiment is designed based on state-of-the-art experiments for tool image classification, see Section 3.4. The state of the art is analyzed in the course of a literature review following the methodology of Webster and Watson 2002, see Section 3.3.

In the course of the literature review conducted by this paper, no paper about neural tool image classification was found. To the best of the knowledge of this paper, this paper is the first paper to examine neural classification of tool images. The practical implications of this are presented in Section 6.9. The training of machine learning algorithms requires data.[182] Therefore, this paper hopes to foster further research in the field of computer vision by providing the TIC Dataset publicly available under the Creative Commons Attribution Share Alike 4.0 International License.

## 6.7 Reflection of Limitations

Time and resources of this paper are limited. Due to these limitations, metalearning, non-neural networks, other computer vision tasks, unsupervised learning, semi-supervised learning, and learning auxiliaries, i.e., transfer learning, adversarial training, data augmentation, input normalization, weight decay, and multi-task learning, are excluded from this paper.

As stated in Section 1.1, an augmented reality solution for field workers requires software perceiving the environment of field workers. This includes other computer vision tasks as well. This paper is limited to a sub-field, i.e., tool image classification.

The scope of this paper is to determine the best-performing neural network for tool image classification. Although neural networks are the state of the art of image classification, as image classification leaderboards show, see Section A.2, it is possible that non-neural networks might be suitable for tool image classification as well.

In the experiment conducted by this paper, neural networks are trained exclusively supervised without learning auxiliaries. Supervised learning can only utilize labeled training data, while semi- and unsupervised learning can utilize unlabeled data as well. More training data and the excluded learning auxiliaries can improve performance.[183] In consequence, the results of the experiment might be improved by implementing those methods.

[182] Pan and Q. Yang 2010; Szegedy, Zaremba, et al. 2014; El-Amir and Hamdy 2020.
[183] Pan and Q. Yang 2010; Szegedy, Zaremba, et al. 2014; El-Amir and Hamdy 2020.

Furthermore, in the experiment, due to limited time and resources, conducting a neural network architecture search and hyperparameter search is excluded. A method to learn neural network architectures and hyperparameters is metalearning.[184] Furthermore, this paper selects the state-of-the-art neural networks determined in the literature review of this paper to conduct the experiment. The literature review regards only the underlying neural network without auxiliaries. If several versions of the underlying neural network exist, the literature review regards the best-performing version according to the paper originally proposing the neural network. Thus, neural architecture search, hyperparameter search, and regarding the different versions of the state-of-the-art neural networks for image classification might reveal neural networks with which it would be worth conducting the experiment. Conducting the experiment with these neural networks might reveal even better-performing neural networks for tool image classification than found in the results of this paper. However, conducting the experiment for these neural networks requires more time and resources than available to this paper.

As discussed in this chapter, EfficientNet-B7 did not learn to classify tool images. On that account, the accuracy of EfficientNet-B7 reported in Chapter 5 cannot be interpreted as the performance of EfficientNet-B7 for tool image classification, but simply as that EfficientNet-B7 was not able to learn in the course of the experiment.

Finally, more training data can further improve performance.[185] Hence, increasing the size of the TIC Dataset might improve the accuracies reported in Chapter 5. Furthermore, tool image classification comprises all classes of tools. The TIC Dataset comprises only classes of tools that were available to this paper, i.e., drill, hammer, pliers, saw, screwdriver, and wrench. The TIC Dataset can be extended by creating more images as described in Section 3.2 or adding subsets containing tool images of already existing datasets.

## 6.8 Future Work

This paper proposes future work in regard to the limitations of this paper. As stated in Section 1.1, an augmented reality solution for field workers requires software perceiving the environment of field workers. Therefore, this paper proposes to investigate the implementation of a holistic solution including other computer vision tasks, a software framework, and a hardware framework.

As discussed in Section 6.7, semi-supervised learning, unsupervised learning, and learning auxiliaries, i.e., transfer learning, adversarial training, data augmentation, input

[184] Schaul and Schmidhuber 2010.
[185] El-Amir and Hamdy 2020.

normalization, weight decay, and multi-task learning, might improve performance. On that account, this paper proposes to investigate the effect of each of those techniques.

As discussed in Section 6.7, regarding neural architecture search, hyperparameter search, different versions of the state-of-the-art neural networks for image classification, and non-neural models to conduct the experiment might reveal even better-performing models for tool image classification than found in the results of this paper. Hence, this paper proposes to investigate these models.

As discussed in Section 6.7, the TIC Dataset can be extended. To train machine learning algorithms, training data is required.[186] As a result, extending the TIC Dataset might help fostering future research in the field of computer vision. Thus, this paper proposes to extend the TIC Dataset by creating more images as described in Section 3.2, adding subsets containing tool images of already existing datasets, and adding additional annotations for object detection, image segmentation or other computer vision tasks.

In the experiment conducted by this paper, EfficientNet-B7 did not learn to classify tool images. As discussed, the accuracy of EfficientNet-B7 reported in Chapter 5 does not reflect the performance of EfficientNet-B7 for tool image classification. As stated in Section 6.5, this paper hypothesizes that the small batch size is the cause. Accordingly, this paper proposes to repeat the experiment on hardware capable of supporting a larger batch size for EfficientNet-B7. If this proves to be insufficient, this paper proposes to reduce regularization as well.

## 6.9 Practical Implications

For industry aiming to implement augmented reality solutions for field workers, the results of the literature review conducted by this paper and the implementation of the neural networks may be of special interest. Implementing augmented reality solutions for field workers requires software perceiving the environment of field workers. The environment can be perceived through computer vision. The results of the literature review conducted by this paper comprise common concepts of neural networks and state-of-the-art neural networks for image classification. These concepts are shared across a variety of neural networks for different computer vision tasks, for example, semantic segmentation, object detection, image generation, pose estimation, and facial recognition.[187] The state-of-the-art neural networks for image classification implement these concepts. To conduct the experiment, these neural networks are implemented.

[186] El-Amir and Hamdy 2020.

[187] Yuan, Xilin Chen, and J. Wang 2019; Tan, Pang, and Q. V. Le 2019; Karras et al. 2018; Bulat et al. 2020; M. Yan et al. 2019.

The implementations are appended in the digital appendix, see Section A.6. As a result, these concepts and implementations can be used as a basis for implementing software perceiving the environment of field workers and, thus, as a basis for implementing augmented reality solutions for field workers. Furthermore, this paper can be seen as proof of concept for software perceiving the environment of field workers.

# 7 Conclusion

In the context of the emerging market for augmented reality solutions for field workers,[188] this paper investigated a sub-task of perceiving the environment of field workers, i.e., tool image classification. Tool image classification was investigated by determining the best-performing neural network for tool image classification in the course of an experiment. This paper found that, in general, several neural networks are suitable for tool image classification. Especially, ResNet-152, ResNeXt-101, and DenseNet-264 were proven to be suitable for tool image classification. For industry, the resulting concept of the literature review conducted by this paper and their implementations can be used as a basis for implementing software perceiving the environment of field workers and, thus, as a basis for implementing augmented reality solutions for field workers. Furthermore, this paper introduced a novel tool image classification dataset called TIC Dataset. This paper hopes to foster further research in the field of computer vision by providing the TIC Dataset publicly available under the Creative Commons Attribution Share Alike 4.0 International License.

[188] Ernst & Young 2019a; Ernst & Young 2019b; Detzel et al. 2018; Shook and Knickrehm 2017; Guy et al. 2019.

# References


Afshar, Parnian, Konstantinos N. Plataniotis, and Arash Mohammadi (2018). "Capsule Networks for Brain Tumor Classification based on MRI Images and Course Tumor Boundaries". In: *CoRR* abs/1811.00597.

Ahmed, Karim and Lorenzo Torresani (2019). "STAR-Caps: Capsule Networks with Straight-Through Attentive Routing". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc, pp. 9101–9110. URL: http://papers.nips.cc/paper/9110-star-caps-capsule-networks-with-straight-through-attentive-routing.pdf.

AlAfandy, Khalid A. et al. (2019). "Artificial Neural Networks Optimization and Convolution Neural Networks to Classifying Images in Remote Sensing". In: *Proceedings of the 4th International Conference on Big Data and Internet of Things*. Ed. by Mohamed Lazaar et al. New York, NY, USA: ACM, pp. 1–8. ISBN: 9781450372404. DOI: 10.1145/3372938.3372945.

Alom, Md Zahangir et al. (2018). "Breast Cancer Classification from Histopathological Images with Inception Recurrent Residual Convolutional Neural Network". In: *CoRR* abs/1811.04241. URL: http://arxiv.org/pdf/1811.04241v1.

Amazon Web Services, Inc. (2020a). *Amazon EC2 G4 Instances*. URL: https://aws.amazon.com/de/ec2/instance-types/g4/.

Amazon Web Services, Inc. (2020b). *AWS Deep Learning AMI (Ubuntu 18.04)*. URL: https://aws.amazon.com/marketplace/pp/Amazon-Web-Services-AWS-Deep-Learning-AMI-Ubuntu-1/B07Y43P7X5.

El-Amir, Hisham and Mahmoud Hamdy (2020). *Deep Learning Pipeline: Building a Deep Learning Model with TensorFlow*. 1st ed. 2020. ISBN: 978-1-4842-5349-6. URL: https://doi.org/10.1007/978-1-4842-5349-6.

Araújo, Teresa et al. (2017). "Classification of breast cancer histology images using Convolutional Neural Networks". In: *PloS one* 12.6, e0177544. DOI: 10.1371/journal.pone.0177544.

Assiri, Yahia Saeed (2019). "Stochastic Optimization of Plain Convolutional Neural Networks with Simple Methods". In: *MLDM*. URL: https://arxiv.org/abs/2001.08856.

Babaie, Morteza et al. (2017). "Classification and Retrieval of Digital Pathology Scans: A New Dataset". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Ed. by IEEE. IEEE, pp. 760–768. ISBN: 978-1-5386-0733-6. DOI: 10.1109/CVPRW.2017.106.

Bansal, Dipali and Rashima Mahajan (2019). "Chapter 2 - EEG-Based Brain-Computer Interfacing (BCI)". In: *EEG-Based Brain-Computer Interfaces*. Ed. by Dipali Bansal and Rashima Mahajan. Academic Press, pp. 21–71. ISBN: 978-0-12-814687-3. DOI: 10.1016/B978-0-12-814687-3.00002-8. URL: http://www.sciencedirect.com/science/article/pii/B9780128146873000028.

Basha, S. ShabbeerH. et al. (2020). "Impact of fully connected layers on performance of convolutional neural networks for image classification". In: *Neurocomputing* 378, pp. 112–119. ISSN: 09252312. DOI: 10.1016/j.neucom.2019.10.008.

Bejnordi, Babak Ehteshami et al. (2017). "Context-aware stacked convolutional neural networks for classification of breast carcinomas in whole-slide histopathology images". In: *Journal of medical imaging (Bellingham, Wash.)* 4.4, p. 044504. ISSN: 2329-4302. DOI: 10.1117/1.JMI.4.4.044504.

Belilovsky, Eugene, Michael Eickenberg, and Edouard Oyallon (2019). "Greedy Layerwise Learning Can Scale To ImageNet". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, pp. 583–593. URL: http://proceedings.mlr.press/v97/belilovsky19a.html.

Beluch, William H. et al. (2018). "The Power of Ensembles for Active Learning in Image Classification". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ed. by IEEE.

Bentes, Carlos, Domenico Velotto, and Bjorn Tings (2018). "Ship Classification in TerraSAR-X Images With Convolutional Neural Networks". In: *IEEE Journal of Oceanic Engineering* 43.1, pp. 258–266. ISSN: 0364-9059. DOI: 10.1109/JOE.2017.2767106.

Berthelot, David et al. (2019). "MixMatch: A Holistic Approach to Semi-Supervised Learning". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc, pp. 5049–5059. URL: http://papers.nips.cc/paper/8749-mixmatch-a-holistic-approach-to-semi-supervised-learning.pdf.

Black, Paul E. (2016). "array". In: *Dictionary of Algorithms and Data Structures*. National Institute of Standards and Technology. URL: https://xlinux.nist.gov/dads/HTML/array.html.

Blot, Michael, Matthieu Cord, and Nicolas Thome (2016). "Maxmin convolutional neural networks for image classification". In: *CoRR* abs/1610.07882.

Bonner, Paul (2018). *The Honeywell Connected Plant*. URL: https://www.honeywellprocess.com/library/news-and-events/presentations/hug-america-2018-honeywell-connected-plant-introduction.pdf (visited on ).

Bossard, Lukas, Matthieu Guillaumin, and Luc Van Gool (2014). "Food-101 – Mining Discriminative Components with Random Forests". In: *European Conference on Computer Vision*.

Bulat, Adrian et al. (2020). "Toward fast and accurate human pose estimation via soft-gated skip connections". In: *ArXiv* abs/2002.11098.

Byerly, Adam, Tatiana Kalganova, and Ian Dear (2020). "A Branching and Merging Convolutional Network with Homogeneous Filter Capsules". In: *ArXiv* abs/2001.09136. URL: https://arxiv.org/abs/2001.09136v3.

Cai, Han, Ligeng Zhu, and Song Han (2018). "ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware". In: *ICLR 2018*. URL: https://arxiv.org/abs/1812.00332v2.

Caruana, Rich (1997). "Multitask Learning". In: *Machine Learning* 28.1, pp. 41–75. ISSN: 08856125. DOI: 10.1023/A:1007379606734.

Castelluccio, Marco et al. (2015). "Land Use Classification in Remote Sensing Images by Convolutional Neural Networks". In: *CoRR* abs/1508.00092.

Chagas, Paulo et al. (2018). "Evaluation of Convolutional Neural Network Architectures for Chart Image Classification". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. Piscataway, NJ, USA: IEEE, pp. 1–8. ISBN: 978-1-5090-6014-6. DOI: 10.1109/IJCNN.2018.8489315.

Chang, Jia-Ren and Yong-Sheng Chen (2015). "Batch-normalized Maxout Network in Network". In: *CoRR* abs/1511.02583. URL: http://arxiv.org/abs/1511.02583.

Chang, Jia-Ren and Yong-Sheng Chen (2017). "Deep Competitive Pathway Networks". In: *CoRR* abs/1709.10282. URL: http://arxiv.org/abs/1709.10282.

Chen, Chaofan et al. (2019). "This Looks Like That: Deep Learning for Interpretable Image Recognition". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc, pp. 8930–8941. URL: http://papers.nips.cc/paper/9095-this-looks-like-that-deep-learning-for-interpretable-image-recognition.pdf.

Chen, Weijie et al. (2019). "All You Need is a Few Shifts: Designing Efficient Convolutional Neural Networks for Image Classification". In: *CoRR* abs/1903.05285.

Chen, Xuan et al. (2018). "Classification of Pancreatic Tumors based on MRI Images using 3D Convolutional Neural Networks". In: *ISICDM 2018*. Ed. by ACM. ICPS. New York, New York: The Association for Computing Machinery, pp. 92–96. ISBN: 9781450365338. DOI: 10.1145/3285996.3286017.

Chen, Zhuo et al. (2020). "ViP: Virtual Pooling for Accelerating CNN-based Image Classification and Object Detection". In: *The IEEE Winter Conference on Applications of Computer Vision (WACV)*. Ed. by IEEE.

Cheng, Gong et al. (2016). "Scene classification of high resolution remote sensing images using convolutional neural networks". In: *2016 IEEE International Geoscience & Remote Sensing Symposium*. Ed. by IEEE. Piscataway, NJ: IEEE, pp. 767–770. ISBN: 978-1-5090-3332-4. DOI: 10.1109/IGARSS.2016.7729193.

Chetlur, Sharan et al. (2014). "cuDNN: Efficient Primitives for Deep Learning". In: *ArXiv* abs/1410.0759.

Chollet, François (2017). "Xception: Deep Learning with Depthwise Separable Convolutions". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807.

Chollet, François et al. (2015a). *Keras*. URL: https://github.com/fchollet/keras.

Chollet, François et al. (2015b). *Keras Applications*. URL: https://github.com/keras-team/keras-applications.

Ciresan, Dan C. et al. (2011). "Flexible, High Performance Convolutional Neural Networks for Image Classification". In: *IJCAI*. Ed. by World Scientific.

Cirean, Dan, Ueli Meier, and Juergen Schmidhuber (2012). "Multi-column Deep Neural Networks for Image Classification". In: *CVPR 2012*. URL: http://arxiv.org/pdf/1202.2745.pdf.

Clanuwat, Tarin et al. (2018). "Deep Learning for Classical Japanese Literature". In: *CoRR* abs/1812.01718. arXiv: 1812.01718. URL: http://arxiv.org/abs/1812.01718.

Coates, Adam, Andrew Ng, and Honglak Lee (2011). "An analysis of single-layer networks in unsupervised feature learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223.

Codella, N. C. F. et al. (2018). "Skin lesion analysis toward melanoma detection: A challenge at the 2017 International symposium on biomedical imaging (ISBI), hosted by the international skin imaging collaboration (ISIC)". In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pp. 168–172.

Cohen, Gregory et al. (2017). "EMNIST: Extending MNIST to handwritten letters". In: *2017 International Joint Conference on Neural Networks (IJCNN)*. DOI: 10.1109/ijcnn.2017.7966217.

Combalia, Marc et al. (2019). *BCN20000: Dermoscopic Lesions in the Wild*. arXiv: 1908.02288 [eess.IV].

Cubuk, Ekin Dogus et al. (2018). "AutoAugment: Learning Augmentation Policies from Data". In: *ArXiv* abs/1805.09501. URL: https://arxiv.org/abs/1805.09501v3.

Dao-Duc, Cuong, Hua Xiaohui, and Olivier Morère (2015). "Maritime Vessel Images Classification Using Deep Convolutional Neural Networks". In: *SoICT 2015*. Ed. by Huynh Quyet Thang et al. ICPS. New York, New York: The Association for Computing Machinery, pp. 1–6. ISBN: 9781450338431. DOI: 10.1145/2833258.2833266.

Darlow, Luke Nicholas et al. (2018). "CINIC-10 is not ImageNet or CIFAR-10". In: *ArXiv* abs/1810.03505.

Das, Arindam et al. (2018). "Document Image Classification with Intra-Domain Transfer Learning and Stacked Generalization of Deep Convolutional Neural Networks". In: *2018 24th International Conference on Pattern Recognition (ICPR)*. Ed. by IEEE. IEEE, pp. 3180–3185. ISBN: 978-1-5386-3788-3. DOI: 10.1109/ICPR.2018.8545630.

de Oliveira Castro, P., S. Louise, and D. Barthou (2010). "A Multidimensional Array Slicing DSL for Stream Programming". In: *2010 International Conference on Complex, Intelligent and Software Intensive Systems*, pp. 913–918.

Debayle, Johan, Nima Hatami, and Yann Gavet (2018). "Classification of time-series images using deep convolutional neural networks". In: *Tenth International Conference on Machine Vision (ICMV 2017)*. Ed. by Antanas Verikas. Proceedings of SPIE. Bellingham, Washington, USA: SPIE, p. 23. ISBN: 9781510619418. DOI: 10.1117/12.2309486. URL: https://spiedigitallibrary.org/conference-proceedings-of-spie/10696/2309486/Classification-of-time-series-images-using-deep-convolutional-neural-networks/10.1117/12.2309486.full.

Deng, J. et al. (2009). "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*.

Denton, Emily, Sam Gross, and Rob Fergus (2016). "Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks". In: *CoRR*. URL: http://arxiv.org/pdf/1611.06430v1.

Detzel, Christopher et al. (2018). *Rolling Out Augmented Reality in the Field*. URL: https://www.bcg.com/de-de/publications/2018/rolling-out-augmented-reality-field.aspx.

Devries, Terrance and Graham W. Taylor (2017). "Improved Regularization of Convolutional Neural Networks with Cutout". In: *ArXiv* abs/1708.04552. URL: https://arxiv.org/pdf/1708.04552v2.pdf.

Dong, Xuanyi et al. (2017). "EraseReLU: A Simple Way to Ease the Training of Deep Convolution Neural Networks". In: *CoRR* abs/1709.07634. URL: http://arxiv.org/abs/1709.07634.

Dvornik, Nikita, Cordelia Schmid, and Julien Mairal (2019). "Diversity With Cooperation: Ensemble Methods for Few-Shot Classification". In: *The IEEE International Conference on Computer Vision (ICCV)*. Ed. by IEEE.

Elfwing, Stefan, Eiji Uchibe, and Kenji Doya (2018). "Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning". In: *Neural networks : the official journal of the International Neural Network Society* 107, pp. 3–11.

Elhoseiny, Mohamed, Sheng Huang, and Ahmed Elgammal (2015). "Weather classification with deep convolutional neural networks". In: *2015 IEEE International Conference on Image Processing (ICIP)*. Ed. by IEEE. Piscataway, NJ: IEEE, pp. 3349–3353. ISBN: 978-1-4799-8339-1. DOI: 10.1109/ICIP.2015.7351424.

Ernst & Young (2019a). *Stop talking about the future of work*. URL: https://assets.ey.com/content/dam/ey-sites/ey-com/en_au/topics/campaigns/future-of-work/stop-talking-about-future-of-work.pdf.

Ernst & Young (2019b). *The Future of Work: the Changing Skills Landscape for Miners*. URL: https://minerals.org.au/sites/default/files/190214%20The%20Future%20of%20Work%20the%20Changing%20Skills%20Landscape%20for%20Miners.pdf.

Ertel, Wolfgang (2016). *Grundkurs künstliche Intelligenz: eine praxisorientierte Einführung*. Springer-Verlag.

Esteva, Andre et al. (2017). "Dermatologist-level classification of skin cancer with deep neural networks". In: *Nature* 542.7639, pp. 115–118. DOI: 10.1038/nature21056.

Fadaeddini, Amin, Mohammad Eshghi, and Babak Majidi (2018). "A deep residual neural network for low altitude remote sensing image classification". In: *2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*. Piscataway, NJ: IEEE, pp. 43–46. ISBN: 978-1-5386-2836-2. DOI: 10.1109/CFIS.2018.8336623.

Fuyong, Xing et al. (2018). "Deep Learning in Microscopy Image Analysis: A Survey". In: *IEEE transactions on neural networks and learning systems* 29.10, pp. 4550–4568. DOI: 10.1109/TNNLS.2017.2766168.

Gad, Ahmed Fawzy (2018). *Practical Computer Vision Applications Using Deep Learning with CNNs: With Detailed Examples in Python Using TensorFlow and Kivy*. Apress. ISBN: 9781484241684. DOI: 10.1007/978-1-4842-4167-7.

Gallego, Antonio-Javier, Antonio Pertusa, and Pablo Gil (2018). "Automatic Ship Classification from Optical Aerial Images with Convolutional Neural Networks". In: *Remote Sensing* 10.4, p. 511. DOI: 10.3390/rs10040511.

Gao, Shanghua et al. (2019). "Res2Net: A New Multi-scale Backbone Architecture". In: *IEEE transactions on pattern analysis and machine intelligence*. DOI: 10.1109/TPAMI.2019.2938758.

Gao, Zhimin et al. (2017). "HEp-2 Cell Image Classification With Deep Convolutional Neural Networks". In: *IEEE journal of biomedical and health informatics* 21.2, pp. 416–428. DOI: 10.1109/JBHI.2016.2526603.

Garcia, Ronald and Andrew Lumsdaine (2005). "MultiArray: a C++ library for generic programming with arrays". In: *Software: Practice and Experience* 35.2, pp. 159–188. ISSN: 0038-0644. DOI: 10.1002/spe.630.

Garud, Hrushikesh et al. (2017). "High-Magnification Multi-Views Based Classification of Breast Fine Needle Aspiration Cytology Cell Samples Using Fusion of Decisions From Deep Convolutional Networks". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. Ed. by IEEE.

Gastaldi, Xavier (2017). "Shake-Shake regularization". In: *ArXiv* abs/1705.07485. URL: https://arxiv.org/abs/1705.07485.

Gong, Chengyue et al. (2020). "MaxUp: A Simple Way to Improve Generalization of Neural Network Training". In: *CoRR*. URL: http://arxiv.org/pdf/2002.09024v1.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Gowal, Sven et al. (2019). "Scalable Verified Training for Provably Robust Image Classification". In: *The IEEE International Conference on Computer Vision (ICCV)*. Ed. by IEEE.

Guo, Tianmei et al. (2017). "Simple convolutional neural network on image classification". In: *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA 2017)*. Ed. by IEEE. Piscataway, NJ: IEEE, pp. 721–724. ISBN: 978-1-5090-3618-9. DOI: 10.1109/ICBDA.2017.8078730.

Guo, Yanhui et al. (2018). "Multiple Convolutional Neural Network for Skin Dermoscopic Image Classification". In: *2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. Ed. by IEEE. [Piscataway, New Jersey]: IEEE, pp. 365–369. ISBN: 978-1-5386-7568-7. DOI: 10.1109/ISSPIT.2018.8642669.

Guo, Yunhui et al. (2019). "Depthwise Convolution Is All You Need for Learning Multiple Visual Domains". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33, pp. 8368–8375. ISSN: 2159-5399. DOI: 10.1609/aaai.v33i01.33018368.

Gurnani, A. et al. (2019). "SAF-BAGE: Salient Approach for Facial Soft-Biometric Classification - Age, Gender, and Facial Expression". In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. Ed. by Winter, pp. 839–847. DOI: 10.1109/WACV.2019.00094.

Guy, Benjamin et al. (2019). *The coming evolution of field operations*. URL: https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/Operations/Our%20Insights/The%20coming%20evolution%20of%20field%20operations/The-coming-evolution-of-field-operations.ashx.

Hahn, Taeyoung, Myeongjang Pyeon, and Gunhee Kim (2019). "Self-Routing Capsule Networks". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc, pp. 7658–7667. URL: http://papers.nips.cc/paper/8982-self-routing-capsule-networks.pdf.

Harangi, Balazs (2018). "Skin lesion classification with ensembles of deep convolutional neural networks". In: *Journal of biomedical informatics* 86, pp. 25–32. DOI: 10.1016/j.jbi.2018.08.006.

Harris, Ethan et al. (2020). "Understanding and Enhancing Mixed Sample Data Augmentation". In: *CoRR*. URL: http://arxiv.org/pdf/2002.12047v1.

HasanPour, Seyyed Hossein et al. (2016). "Lets keep it simple, Using simple architectures to outperform deeper and more complex architectures". In: *CoRR* abs/1608.06037.

He, Kaiming et al. (2015a). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *The IEEE International Conference on Computer Vision (ICCV)*. Ed. by IEEE.

He, Kaiming et al. (2015b). "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 37.9, pp. 1904–1916. DOI: 10.1109/TPAMI.2015.2389824.

He, Kaiming et al. (2016). "Deep Residual Learning for Image Recognition". In: *29th IEEE Conference on Computer Vision and Pattern Recognition*. Ed. by IEEE. Piscataway, NJ: IEEE, pp. 770–778. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.90.

He, Tong et al. (2018). "Bag of Tricks for Image Classification with Convolutional Neural Networks". In: *CoRR* abs/1812.01187. URL: http://arxiv.org/pdf/1812.01187.pdf.

Honeywell International Inc. (2018). *Honeywell Connected Worker Solutionss*. URL: https://www.honeywellaidc.com/solutions/connected-worker (visited on ).

Hori, Kazunori, Shogo Okada, and Katsumi Nitta (2016). "Fashion image classification on mobile phones using layered deep convolutional neural networks". In: *MUM 2016*. Ed. by Florian Alt. ACM International Conference Proceeding Series. New York: ACM, pp. 359–361. ISBN: 9781450348607. DOI: 10.1145/3012709.3016075.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5, pp. 359–366. ISSN: 08936080. DOI: 10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/abs/pii/0893608089900208?via%3Dihub.

Howard, Andrew G. (2013). "Some Improvements on Deep Convolutional Neural Network Based Image Classification". In: *CoRR* abs/1312.5402. URL: https://arxiv.org/abs/1312.5402.

Hu, Jie, Li Shen, and Gang Sun (2018). "Squeeze-and-Excitation Networks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Ed. by IEEE. Piscataway, NJ: IEEE, pp. 7132–7141. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00745.

Huang, Gao et al. (2017). "Densely Connected Convolutional Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ed. by IEEE. IEEE, pp. 2261–2269. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.243.

Huang, Siyuan et al. (2019). "PerspectiveNet: 3D Object Detection from a Single RGB Image via Perspective Points". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc, pp. 8905–8917. URL: http://papers.nips.cc/paper/9093-perspectivenet-3d-object-detection-from-a-single-rgb-image-via-perspective-points.pdf.

Huang, Yanping et al. (2019). "GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc, pp. 103–112. URL: http://papers.nips.cc/paper/8305-gpipe-efficient-training-of-giant-neural-networks-using-pipeline-parallelism.pdf.

Iesmantas, Tomas and Robertas Alzbutas (2018). "Convolutional capsule network for classification of breast cancer histology images". In: *CoRR* abs/1804.08376.

Ioannou, Yani et al. (2015). "Training CNNs with Low-Rank Filters for Efficient Image Classification". In: *CoRR* abs/1511.06744.

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html.

Izmailov, Pavel et al. (2018). "Averaging Weights Leads to Wider Optima and Better Generalization". In: *UAI*. Ed. by AUAI.

Jayasundara, Vinoj et al. (2019). "TextCaps: Handwritten Character Recognition With Very Small Datasets". In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 254–262. URL: https://arxiv.org/abs/1904.08095v1.

Jeon, Yunho and Junmo Kim (2017). "Active Convolution: Learning the Shape of Convolution for Image Classification". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ed. by IEEE.

Ji, Xu, Andrea Vedaldi, and Joao Henriques (2019). "Invariant Information Clustering for Unsupervised Image Classification and Segmentation". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Ed. by IEEE. IEEE, pp. 9864–9873. ISBN: 978-1-7281-4803-8. DOI: 10.1109/ICCV.2019.00996.

Jiménez-Sánchez, Amelia, Shadi Albarqouni, and Diana Mateus (2018). "Capsule Networks Against Medical Imaging Data Challenges". In: *Intravascular imaging and computer assisted stenting and large-scale annotation of biomedical data and expert label synthesis*. Ed. by Danail Stoyanov et al. Vol. 11043. Lecture Notes in Computer Science. Cham: Springer, pp. 150–160. ISBN: 978-3-030-01363-9. DOI: 10.1007/978-3-030-01364-617.

Ju, Cheng, Aurélien Bibaut, and Mark van der Laan (2018). "The Relative Performance of Ensemble Methods with Deep Convolutional Neural Networks for Image Classification". In: *Journal of applied statistics* 45.15, pp. 2800–2818. ISSN: 0266-4763. DOI: 10.1080/02664763.2018.1441383.

Kapur, Saurabh (2017). *Computer vision with Python 3: Image classification, object detection, video processing, and more*. Birmingham, UK: Packt Publishing. ISBN: 9781788299763.

Karras, Tero et al. (2018). "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *ArXiv* abs/1710.10196.

Kesim, Ege, Zumray Dokur, and Tamer Olmez (2019). "X-Ray Chest Image Classification by A Small-Sized Convolutional Neural Network". In: *2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)*. Ed. by IEEE. IEEE, pp. 1–5. ISBN: 978-1-7281-1013-4. DOI: 10.1109/EBBT.2019.8742050.

Kieffer, Brady et al. (2017). "Convolutional neural networks for histopathology image classification: Training vs. Using pre-trained networks". In: *Proceedings of the Seventh International Conference on Image Processing Theory, Tools and Applications - IPTA 2017*. Piscataway, NJ: IEEE, pp. 1–6. ISBN: 978-1-5386-1842-4. DOI: 10.1109/IPTA.2017.8310149.

Kim, Pyong-Kun and Kil-Taek Lim (2017). "Vehicle Type Classification Using Bagging and Convolutional Neural Network on Multi View Surveillance Image". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Ed. by IEEE. IEEE, pp. 914–919. ISBN: 978-1-5386-0733-6. DOI: 10.1109/CVPRW.2017.126.

Kolesnikov, Alexander I. et al. (2019). "Large Scale Learning of General Visual Representations for Transfer". In: *ArXiv* abs/1912.11370.

Kortylewski, Adam et al. (2020). "Combining Compositional Models and Deep Networks For Robust Object Classification under Occlusion". In: *The IEEE Winter Conference on Applications of Computer Vision (WACV)*. Ed. by IEEE.

Kosiorek, Adam et al. (2019). "Stacked Capsule Autoencoders". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc, pp. 15512–15522. URL: http://papers.nips.cc/paper/9684-stacked-capsule-autoencoders.pdf.

Kowsari, Kamran et al. (2018). "RMDL". In: *ICISDM 2018 2nd International Conference on Information System and Data Mining : Florida Polytechnic University, Florida, USA, April 9-11, 2018*. Ed. by Association for Computing Machinery. ICPS. New York, New York: The Association for Computing Machinery, pp. 19–28. ISBN: 9781450363549. DOI: 10.1145/3206098.3206111.

Krause, Jonathan et al. (2013). "3D Object Representations for Fine-Grained Categorization". In: *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia.

Krizhevsky, Alex (May 2012). "Learning Multiple Layers of Features from Tiny Images". In: *University of Toronto*.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2017). "ImageNet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6, pp. 84–90. ISSN: 00010782. DOI: 10.1145/3065386.

Kubilius, Jonas et al. (2019). "Brain-Like Object Recognition with High-Performing Shallow Recurrent ANNs". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc, pp. 12805–12816. URL: http://papers.nips.cc/paper/9441-brain-like-object-recognition-with-high-performing-shallow-recurrent-anns.pdf.

Kyrkou, Christos and Theocharis Theocharides (2019). "Deep-Learning-Based Aerial Image Classification for Emergency Response Applications Using Unmanned Aerial Vehicles". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. Ed. by IEEE.

Law, Stephen, Yao Shen, and Chanuki Seresinhe (2017). "An application of convolutional neural network in street image classification". In: *Proceedings of the 1st Workshop on GeoAI*. Ed. by Huina Mao et al. New York, New York: The Association for Computing Machinery, pp. 5–9. ISBN: 9781450354981. DOI: 10.1145/3149808.3149810.

Le Hou et al. (2015). "Patch-based Convolutional Neural Network for Whole Slide Tissue Image Classification". In: *CoRR* abs/1504.07947. URL: http://arxiv.org/pdf/1504.07947.pdf.

Lecun, Y. et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324. ISSN: 00189219. DOI: 10.1109/5.726791.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015a). "Deep learning". In: *Nature* 521.7553, pp. 436–444. DOI: 10.1038/nature14539.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015b). "Deep learning". In: *Nature* 521.7553, pp. 436–444. DOI: 10.1038/nature14539. URL: https://www.nature.com/articles/nature14539.

LeCun, Yann, Corinna Cortes, and CJ Burges (2010). "MNIST handwritten digit database". In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2.

Lenssen, Jan Eric, Matthias Fey, and Pascal Libuschewski (2018). "Group Equivariant Capsule Networks". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc, pp. 8844–8853. URL: http://papers.nips.cc/paper/8100-group-equivariant-capsule-networks.pdf.

Leroux, Sam et al. (2018). "IamNN: Iterative and Adaptive Mobile Neural Network for efficient image classification". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. Ed. by ICLR. OpenReview.net. URL: https://arxiv.org/abs/1804.10123.

Levi, Gil and Tal Hassncer (2015). "Age and gender classification using convolutional neural networks". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition workshops (CVPRW)*. Ed. by IEEE. Piscataway, NJ: IEEE, pp. 34–42. ISBN: 978-1-4673-6759-2. DOI: 10.1109/CVPRW.2015.7301352.

Li, Bin et al. (2018). "Benign and malignant mammographic image classification based on Convolutional Neural Networks". In: *Proceedings of 2018 10th International Conference on Machine Learning and Computing (ICMLC 2018)*. Ed. by ACM. ICPS. New York, New York, USA: The Association for Computing Machinery, pp. 247–251. ISBN: 978-1-4503-6353-2.

Lian, Chunyan et al. (2018). "Deep Convolutional Neural Networks for Diabetic Retinopathy Classification". In: *ICAIP 2018*. Ed. by ACM. ICPS. New York, New York: The Association for Computing Machinery, pp. 68–72. ISBN: 9781450364607. DOI: 10.1145/3239576.3239589.

Liang, Senwei, Yuehaw Khoo, and Haizhao Yang (2018). "Drop-Activation: Implicit Parameter Reduction and Harmonic Regularization". In: *CoRR* abs/1811.05850. URL: http://arxiv.org/abs/1811.05850.

Lim, Sungbin et al. (2019). "Fast AutoAugment". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc, pp. 6665–6675. URL: http://papers.nips.cc/paper/8892-fast-autoaugment.pdf.

Lin, Min, Qiang Chen, and Shuicheng Yan (2013). "Network In Network". In: *CoRR* abs/1312.4400. URL: https://arxiv.org/abs/1312.4400.

Liu, Baoyuan et al. (2015). "Sparse Convolutional Neural Networks". In: *The IEEE International Conference on Computer Vision (ICCV)*. Ed. by IEEE.

Liu, Chenxi et al. (2018). "Progressive Neural Architecture Search". In: *Computer Vision - ECCV 2018*. Ed. by Vittorio Ferrari et al. Vol. 11205. Image Processing, Computer Vision, Pattern Recognition, and Graphics. Cham: Springer International Publishing, pp. 19–35. ISBN: 978-3-030-01245-8. DOI: 10.1007/978-3-030-01246-52.

Liu, Jiang et al. (2016). "Two-Stream Contextualized CNN for Fine-Grained Image Classification". In: *Thirtieth AAAI Conference on Artificial Intelligence*. Ed. by AAAI Press, pp. 4232–3233. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/viewPaper/11772.

Liu, Qun et al. (2020). "DeepSat V2: feature augmented convolutional neural nets for satellite image classification". In: *Remote Sensing Letters* 11.2, pp. 156–165. ISSN: 2150-704X. DOI: 10.1080/2150704X.2019.1693071.

Liu, Ziwei et al. (2016). "DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lu, Alex et al. (2019). "The Cells Out of Sample (COOS) dataset and benchmarks for measuring out-of-sample generalization of image classifiers". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc, pp. 1854–1862. URL: http://papers.nips.cc/paper/8461-the-cells-out-of-sample-coos-dataset-and-benchmarks-for-measuring-out-of-sample-generalization-of-image-classifiers.pdf.

Madrazo, Celia Fernández et al. (2019). "Application of a Convolutional Neural Network for image classification for the analysis of collisions in High Energy Physics". In: *EPJ Web of Conferences* 214, p. 06017. DOI: 10.1051/epjconf/201921406017.

Mahajan, Dhruv et al. (2018). "Exploring the Limits of Weakly Supervised Pretraining". In: *Computer Vision - ECCV 2018*. Ed. by Vittorio Ferrari et al. Vol. 11206. Image Processing, Computer Vision, Pattern Recognition, and Graphics. Cham: Springer International Publishing, pp. 185–201. ISBN: 978-3-030-01215-1. DOI: 10.1007/978-3-030-01216-812.

Martn Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Matsunaga, Kazuhisa et al. (2017). "Image Classification of Melanoma, Nevus and Seborrheic Keratosis by Deep Neural Network Ensemble". In: *CoRR* abs/1703.03108. URL: http://arxiv.org/pdf/1703.03108v1.

McDonnell, Mark D. and Tony Vladusich (2015). "Enhanced Image Classification With a Fast-Learning Shallow Convolutional Neural Network". In: *CoRR* abs/1503.04596. URL: http://arxiv.org/pdf/1503.04596v3.

Mendes, Danilo Barros and Nilton Correia da Silva (2018). "Skin Lesions Classification Using Convolutional Neural Networks in Clinical Images". In: *CoRR* abs/1812.02316.

Michelucci, Umberto (2019). *Advanced applied deep learning: Convolutional neural networks and object detection*. New York: Apress. ISBN: 978-1-4842-4975-8.

Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar (2012). *Foundations of machine learning*. Adaptive computation and machine learning. Cambridge, Mass. and London: The MIT Press. ISBN: 9780262018258.

Murthy, Venkatesh N. et al. (2016). "Deep Decision Network for Multi-Class Image Classification". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ed. by IEEE.

Nazeri, Kamyar, Azad Aminpour, and Mehran Ebrahimi (2018). "Two-Stage Convolutional Neural Network for Breast Cancer Histology Image Classification". In: *Image analysis and recognition15th international conference, ICIAR 2018, Póvoa de Varzim, Portugal, June 27-29, 2018 : proceedings*. Ed. by Aurélio Campilho, Fakhri Karray, and Bart M. ter Haar Romeny. Lecture Notes in Computer Science. Cham: Springer. ISBN: 978-3-319-92999-6.

Nejad, Elaheh Mahraban et al. (2017). "Classification of Histopathology Images of Breast into Benign and Malignant using a Single-layer Convolutional Neural Network". In: *Proceedings of the International Conference on Imaging, Signal Processing and Communication*. Ed. by ACM. [Place of publication not identified]: ACM, pp. 50–53. ISBN: 9781450352895. DOI: 10.1145/3132300.3132331.

Netzer, Yuval et al. (2011). "Reading Digits in Natural Images with Unsupervised Feature Learning". In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. URL: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Nilsback, M-E. and A. Zisserman (2008). "Automated Flower Classification over a Large Number of Classes". In: *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*.

Nogueira, Keiller, Otávio A.B. Penatti, and Jefersson A. dos Santos (2017). "Towards better exploiting convolutional neural networks for remote sensing scene classification". In: *Pattern Recognition* 61, pp. 539–556. ISSN: 00313203. DOI: 10.1016/j.patcog.2016.07.001.

Nøkland, Arild and Lars Hiller Eidnes (2019). "Training Neural Networks with Local Error Signals". In: *ArXiv* abs/1901.06656. URL: https://arxiv.org/abs/1901.06656v2.

NVIDEA (2007). *CUDA Technology*. URL: https://developer.nvidia.com/cuda-toolkit.

Orhan, Emin (2018). "A Simple Cache Model for Image Recognition". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc, pp. 10107–10116. URL: http://papers.nips.cc/paper/8214-a-simple-cache-model-for-image-recognition.pdf.

Pan, Sinno Jialin and Qiang Yang (2010). "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10, pp. 1345–1359. ISSN: 1041-4347. DOI: 10.1109/TKDE.2009.191. URL: https://www.cse.ust.hk/~qyang/Docs/2009/tkde_transfer_learning.pdf.

Peng, Binbin et al. (2018). "Fully Convolutional Neural Networks for Tissue Histopathology Image Classification and Segmentation". In: *2018 IEEE International Conference on Image Processing*. Piscataway, NJ: IEEE, pp. 1403–1407. ISBN: 978-1-4799-7061-2. DOI: 10.1109/ICIP.2018.8451517.

Perez, Fabio, Sandra Avila, and Eduardo Valle (2019). "Solo or Ensemble? Choosing a CNN Architecture for Melanoma Classification". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. Ed. by IEEE.

Phan, Hai et al. (2020). "MoBiNet: A Mobile Binary Network for Image Classification". In: *The IEEE Winter Conference on Applications of Computer Vision (WACV)*. Ed. by IEEE.

Pimkin, Artem et al. (2018). "Ensembling Neural Networks for Digital Pathology Images Classification and Segmentation". In: *Image analysis and recognition15th international conference, ICIAR 2018, Póvoa de Varzim, Portugal, June 27-29, 2018 : proceedings*. Ed. by Aurélio Campilho, Fakhri Karray, and Bart M. ter Haar Romeny. Vol. 10882. Lecture Notes in Computer Science. Cham: Springer, pp. 877–886. ISBN: 978-3-319-92999-6. DOI: 10.1007/978-3-319-93000-8100.

Ponti, Moacir Antonelli et al. (2017). "Everything You Wanted to Know about Deep Learning for Computer Vision but Were Afraid to Ask". In: *SIBGRAPI-T 2017*. Piscataway, NJ: IEEE, pp. 17–41. ISBN: 978-1-5386-0619-3. DOI: 10.1109/SIBGRAPI-T.2017.12.

Ramachandran, Prajit, Barret Zoph, and Quoc V. Le (2017). "Swish: a Self-Gated Activation Function". In: *arXiv: Neural and Evolutionary Computing*.

Rastegari, Mohammad et al. (2016). "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks". In: *Computer vision - ECCV 2016*. Ed. by Bastian Leibe et al. Vol. 9908. Lecture Notes in Computer Science. Cham: Springer, pp. 525–542. ISBN: 978-3-319-46492-3. DOI: 10.1007/978-3-319-46493-032.

Rawat, Waseem and Zenghui Wang (2017). "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review". In: *Neural computation* 29.9, pp. 2352–2449. DOI: 10.1162/NECOa00990.

Real, Esteban et al. (2019). "Regularized Evolution for Image Classifier Architecture Search". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33, pp. 4780–4789. ISSN: 2159-5399. DOI: 10.1609/aaai.v33i01.33014780.

Roth, Holger R. et al. (2015). "Anatomy-specific classification of medical images using deep convolutional nets". In: *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*. Piscataway, NJ: IEEE, pp. 101–104. ISBN: 978-1-4799-2374-8. DOI: 10.1109/ISBI.2015.7163826.

Ruder, Sebastian (2016). "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747*.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning representations by back-propagating errors". In: *Nature* 323.6088, pp. 533–536. DOI: 10.1038/323533a0. URL: https://www.nature.com/articles/323533a0.

Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton (2017). "Dynamic Routing Between Capsules". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc, pp. 3856–3866. URL: http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf.

Al-Saffar, Ahmed Ali Mohammed, Hai Tao, and Mohammed Ahmed Talab (2017). "Review of deep convolution neural network in image classification". In: *2017 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)*. Piscataway, NJ: IEEE, pp. 26–31. ISBN: 978-1-5386-3849-1. DOI: 10.1109/ICRAMET.2017.8253139.

Santurkar, Shibani et al. (2018). "How Does Batch Normalization Help Optimization?" In: *NeurIPS*.

Sato, Ikuro, Hiroki Nishimura, and Kensuke Yokoi (2015). "APAC: Augmented PAttern Classification with Neural Networks". In: *CoRR* abs/1505.03229. URL: http://arxiv.org/abs/1505.03229.

Schaul, Tom and Juergen Schmidhuber (2010). "Metalearning". In: *Scholarpedia* 5.6, p. 4650. DOI: 10.4249/scholarpedia.4650.

Seo, Yian and Kyung-shik Shin (2018). "Image classification of fine-grained fashion image based on style using pre-trained convolutional neural network". In: *2018 IEEE 3rd International Conference on Big Data Analysis (ICBDA 2018)*. Piscataway, NJ: IEEE, pp. 387–390. ISBN: 978-1-5386-4794-3. DOI: 10.1109/ICBDA.2018.8367713.

Shen, Xu et al. (2016). "Transform-Invariant Convolutional Neural Networks for Image Classification and Search". In: *MM'16 Proceedings of the 2016 ACM Multimedia Conference : October 15-19, 2016, Amsterdam, The Netherlands*. Ed. by Alan Hanjalic et al. New York, NY: ACM Association for Computing Machinery, pp. 1345–1354. ISBN: 9781450336031. DOI: 10.1145/2964284.2964316.

Shetty, Suyash (2016). "Application of Convolutional Neural Network for Image Classification on Pascal VOC Challenge 2012 dataset". In: *CoRR* abs/1607.03785. URL: https://arxiv.org/pdf/1607.03785.pdf.

Shi, Weiwei et al. (2019). "Fine-Grained Image Classification Using Modified DCNNs Trained by Cascaded Softmax and Generalized Large-Margin Losses". In: *IEEE transactions on neural networks and learning systems* 30.3, pp. 683–694. DOI: 10.1109/TNNLS.2018.2852721.

Shima, Yoshihiro and Yuki Omori (2018). "Image Augmentation for Classifying Facial Expression Images by Using Deep Neural Network Pre-trained with Object Image Database". In: *Proceedings of ICRCA 2018*. Ed. by ACM. ICPS. New York, New York: The Association for Computing Machinery, pp. 140–146. ISBN: 9781450365307. DOI: 10.1145/3265639.3265664.

Shook, Ellyn and Mark Knickrehm (2017). *Harnessing Revolution Creating the future workforce*. URL: https://www.accenture.com/_acnmedia/pdf-40/accenture-strategy-harnessing-revolution-pov.pdf.

Simonyan, Karen and Andrew Zisserman (2014). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556. URL: https://arxiv.org/abs/1409.1556.

Singh, Bharat et al. (2018). "R-FCN-3000 at 30fps: Decoupling Detection and Classification". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ed. by IEEE.

Singh, Pramod and Avinash Manure (2020). *Learn TensorFlow 2.0: Implement Machine Learning and Deep Learning Models with Python*. 1st ed. 2020. ISBN: 978-1-4842-5558-2. URL: https://doi.org/10.1007/978-1-4842-5558-2.

Sladojevic, Srdjan et al. (2016). "Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification". In: *Computational intelligence and neuroscience* 2016, p. 3289801. DOI: 10.1155/2016/3289801.

Slavkovikj, Viktor et al. (2015). "Hyperspectral Image Classification with Convolutional Neural Networks". In: *MM'15*. Ed. by Xiaofang Zhou et al. New York, New York, USA: ACM Press, pp. 1159–1162. ISBN: 9781450334594. DOI: 10.1145/2733373.2806306.

Sosnovik, Ivan, Micha Szmaja, and Arnold Smeulders (2020). "Scale-Equivariant Steerable Networks". In: *ICLR 2020*. URL: https://openreview.net/pdf?id=HJgpugrKPS.

Spanhol, Fabio Alexandre et al. (2016). "Breast cancer histopathological image classification using Convolutional Neural Networks". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. Ed. by IEEE. Piscataway, NJ: IEEE, pp. 2560–2567. ISBN: 978-1-5090-0620-5. DOI: 10.1109/IJCNN.2016.7727519.

Srivastava, Nitish et al. (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1, pp. 1929–1958.

Stamoulis, Dimitrios et al. (2018). "Designing Adaptive Neural Networks for Energy-Constrained Image Classification". In: *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. Ed. by Iris Bahar. Piscataway, NJ: IEEE, pp. 1–8. ISBN: 9781450359504. DOI: 10.1145/3240765.3240796.

Statista (2019). *In-depth: Industry 4.0 2019 Statista Digital Market Outlook*. URL: https://de.statista.com/statistik/studie/id/67366/dokument/in-depth-industry-40/.

Su, Dong et al. (2018). "Is Robustness the Cost of Accuracy? – A Comprehensive Study on the Robustness of 18 Deep Image Classification Models". In: *The European Conference on Computer Vision (ECCV)*. Ed. by Springer.

Sultana, Farhana, Abu Sufian, and Paramartha Dutta (2019). "Advancements in Image Classification using Convolutional Neural Network". In: *Proceedings, 2018 Fourth IEEE International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*. Ed. by Siddhartha Bhattacharyya. [Piscataway, New Jersey]: IEEE, pp. 122–129. ISBN: 978-1-5386-7638-7. DOI: 10.1109/ICRCICN.2018.8718718.

Sun, Yanan et al. (2019). "Evolving Deep Convolutional Neural Networks for Image Classification". In: *IEEE Transactions on Evolutionary Computation*, p. 1. ISSN: 1089-778X. DOI: 10.1109/TEVC.2019.2916183.

Sun, Zhun, Mete Ozay, and Takayuki Okatani (2016). "Design of Kernels in Convolutional Neural Networks for Image Classification". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Vol. 9911. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 51–66. ISBN: 978-3-319-46477-0. DOI: 10.1007/978-3-319-46478-74.

Szegedy, Christian, Sergey Ioffe, et al. (2017). "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. Ed. by AAAI Press. AAAI'17. AAAI Press, pp. 4278–4284.

Szegedy, Christian, Wei Liu, et al. (2015). "Going deeper with convolutions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ed. by IEEE. IEEE, pp. 1–9. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7298594.

Szegedy, Christian, Vincent Vanhoucke, et al. (2016). "Rethinking the Inception Architecture for Computer Vision". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Szegedy, Christian, Wojciech Zaremba, et al. (2014). "Intriguing properties of neural networks". In: *CoRR* abs/1312.6199.

Taha, Ahmed et al. (2020). "Boosting Standard Classification Architectures Through a Ranking Regularizer". In: *The IEEE Winter Conference on Applications of Computer Vision (WACV)*. Ed. by IEEE.

Tan, Mingxing and Quoc Le V (2019). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *2019 International Conference on Machine Learning (ICML)*. Ed. by ICML. URL: https://arxiv.org/pdf/1905.11946v3.pdf.

Tan, Mingxing, Ruoming Pang, and Quoc V. Le (2019). "EfficientDet: Scalable and Efficient Object Detection". In: *ArXiv* abs/1911.09070.

Tensmeyer, Chris and Tony Martinez (2017). "Analysis of Convolutional Neural Networks for Document Image Classification". In: *CoRR* abs/1708.03273.

Teramoto, Atsushi et al. (2017). "Automated Classification of Lung Cancer Types from Cytological Images Using Deep Convolutional Neural Networks". In: *BioMed research international* 2017. DOI: 10.1155/2017/4067832.

Tokozume, Yuji, Yoshitaka Ushiku, and Tatsuya Harada (2018). "Between-Class Learning for Image Classification". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ed. by IEEE.

Too, Edna C. et al. (2019). "The Convolution Neural Network with Transformed Exponential Linear Unit Activation Function for Image Classification". In: *IVSP 2019*. Ed. by ACM. ICPS. New York, New York: The Association for Computing Machinery, pp. 55–62. ISBN: 9781450361750. DOI: 10.1145/3317640.3317649.

Touvron, Hugo et al. (2019). "Fixing the train-test resolution discrepancy". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc, pp. 8252–8262. URL: http://papers.nips.cc/paper/9035-fixing-the-train-test-resolution-discrepancy.pdf.

Tschandl, Philipp, Cliff Rosendahl, and Harald Kittler (2018). "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions". In: *Scientific data* 5, p. 180161. DOI: 10.1038/sdata.2018.161.

Tyas, Dyah Aruming et al. (2017). "The Classification of Abnormal Red Blood Cell on The Minor Thalassemia Case Using Artificial Neural Network and Convolutional Neural Network". In: *Proceedings of the International Conference on Video and Image Processing*. Ed. by ACM. [Place of publication not identified]: ACM, pp. 228–233. ISBN: 9781450353830. DOI: 10.1145/3177404.3177438.

Vaila, Ruthvik, John Chiasson, and Vishal Saxena (2019). "Deep Convolutional Spiking Neural Networks for Image Classification". In: *CoRR* abs/1903.12272.

van Horn, Grant et al. (2018). "The INaturalist Species Classification and Detection Dataset". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ed. by IEEE.

Van Rossum, Guido and Fred L. Drake (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace. ISBN: 1441412697.

Vo, An Tien, Hai Son Tran, and Thai Hoang Le (2017). "Advertisement image classification using convolutional neural network". In: *KSE 2017*. Ed. by Thanh-Thuy Nguyen. Piscataway, NJ: IEEE, pp. 197–202. ISBN: 978-1-5386-3576-6. DOI: 10.1109/KSE.2017.8119458.

Wah, C. et al. (2011). *The Caltech-UCSD Birds-200-2011 Dataset*. Tech. rep. CNS-TR-2011-001. California Institute of Technology.

Wan, Li et al. (2013). "Regularization of Neural Networks using DropConnect". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research. Atlanta, Georgia, USA: PMLR, pp. 1058–1066. URL: http://proceedings.mlr.press/v28/wan13.html.

Wang, Bin et al. (2018). "A Hybrid Differential Evolution Approach to Designing Deep Convolutional Neural Networks for Image Classification". In: *AI 2018: Advances in Artificial Intelligence*. Ed. by Tanja Mitrovic, Bing Xue, and Xiaodong Li. Vol. 11320. Lecture Notes in Artificial Intelligence. Cham: Springer International Publishing, pp. 237–250. ISBN: 978-3-030-03990-5. DOI: 10.1007/978-3-030-03991-224.

Wang, Fei et al. (2017). "Residual Attention Network for Image Classification". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ed. by IEEE.

Wang, Robert J., Xiang Li, and Charles X. Ling (2018). "Pelee: A Real-Time Object Detection System on Mobile Devices". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc, pp. 1963–1972. URL: http://papers.nips.cc/paper/7466-pelee-a-real-time-object-detection-system-on-mobile-devices.pdf.

Wang, Xiaosong et al. (2017). "ChestX-ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ed. by IEEE.

Wang, Xiao et al. (2019). "EnAET: Self-Trained Ensemble AutoEncoding Transformations for Semi-Supervised Learning". In: *ArXiv* abs/1911.09265. URL: https://arxiv.org/abs/1911.09265v1.

Wang, Yan et al. (2018). "Multi-Scale Spatially-Asymmetric Recalibration for Image Classification". In: *The European Conference on Computer Vision (ECCV)*. Ed. by Springer.

Wang, Yiru et al. (2019). "Dynamic Curriculum Learning for Imbalanced Data Classification". In: *The IEEE International Conference on Computer Vision (ICCV)*. Ed. by IEEE.

Webster, Jane and Richard T. Watson (2002). "Analyzing the Past to Prepare for the Future: Writing a Literature Review". In: *MIS Quarterly* 26.2, pp. xiii–xxiii. ISSN: 02767783. URL: http://www.jstor.org/stable/4132319.

Wu, Ruobing et al. (2015). "Harvesting Discriminative Meta Objects With Deep CNN Features for Scene Classification". In: *The IEEE International Conference on Computer Vision (ICCV)*. Ed. by IEEE.

Xiao, Han, Kashif Rasul, and Roland Vollgraf (2017). "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". In: *CoRR* abs/1708.07747. arXiv: 1708.07747. URL: http://arxiv.org/abs/1708.07747.

Xiao, Tianjun et al. (2015). "The Application of Two-Level Attention Models in Deep Convolutional Neural Network for Fine-Grained Image Classification". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ed. by IEEE.

Xie, Cihang et al. (2019). "Adversarial Examples Improve Image Recognition". In: *CoRR*. URL: http://arxiv.org/pdf/1911.09665v1.

Xie, Qizhe et al. (2019). "Self-training with Noisy Student improves ImageNet classification". In: *ArXiv* abs/1911.04252. URL: https://arxiv.org/abs/1911.04252v2.

Xie, Saining et al. (2017). "Aggregated Residual Transformations for Deep Neural Networks". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xie, Yiting and David Richmond (2019). "Pre-training on Grayscale ImageNet Improves Medical Image Classification". In: *Computer vision - ECCV 2018 workshops*. Ed. by Laura Leal-Taixé and Stefan Roth. Vol. 11134. Lecture Notes in Computer Science. Cham: Springer, pp. 476–484. ISBN: 978-3-030-11023-9. DOI: 10.1007/978-3-030-11024-637.

Xiong, Yunyang, Hyunwoo J. Kim, and Varsha Hedau (2019). "ANTNets: Mobile Convolutional Neural Networks for Resource Efficient Image Classification". In: *CoRR* abs/1904.03775. URL: http://arxiv.org/pdf/1904.03775.pdf.

Xu, Hang et al. (2019). "Auto-FPN: Automatic Network Architecture Adaptation for Object Detection Beyond Classification". In: *The IEEE International Conference on Computer Vision (ICCV)*. Ed. by IEEE.

Xu, Lian et al. (2019). "Coral Classification Using DenseNet and Cross-modality Transfer Learning". In: *IJCNN 2019*. Ed. by IEEE. [Piscataway, New Jersey]: [IEEE], pp. 1–8. ISBN: 978-1-7281-1985-4. DOI: 10.1109/IJCNN.2019.8852235.

Yamada, Yoshihiro et al. (2019). "Shakedrop Regularization for Deep Residual Learning". In: *IEEE Access* 7, pp. 186126–186136. DOI: 10.1109/ACCESS.2019.2960566.

Yan, Mengjia et al. (2019). "VarGFaceNet: An Efficient Variable Group Convolutional Neural Network for Lightweight Face Recognition". In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 2647–2654.

Yang, Hong-Ming et al. (2018). "Robust Classification With Convolutional Prototype Learning". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ed. by IEEE.

Yang, Jufeng, Dongyu She, and Ming Sun (2017). "Joint Image Emotion Classification and Distribution Learning via Deep Convolutional Neural Network". In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. Ed. by Fahiem Bacchus and Carles Sierra. California: International Joint Conferences on Artificial Intelligence Organization, pp. 3266–3272. ISBN: 9780999241103. DOI: 10.24963/ijcai.2017/456.

Yuan, Yuhui, Xilin Chen, and Jingdong Wang (2019). "Object-Contextual Representations for Semantic Segmentation". In: *ArXiv* abs/1909.11065.

Yun, Sangdoo et al. (2019). "CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Ed. by IEEE. IEEE, pp. 6022–6031. ISBN: 978-1-7281-4803-8. DOI: 10.1109/ICCV.2019.00612.

Zagoruyko, Sergey and Nikos Komodakis (2016). "Wide Residual Networks". In: *Procedings of the British Machine Vision Conference 2016*. Ed. by R. C. Wilson et al. British Machine Vision Association, pp. 87.1–87.12. ISBN: 1-901725-59-6. DOI: 10.5244/C.30.87.

Zahavy, Tom et al. (2018). "Is a Picture Worth a Thousand Words? A Deep Multi-Modal Architecture for Product Classification in E-Commerce". In: *AAAI Conference on Artificial Intelligence*. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16579.

Zeiler, Matthew D. and Rob Fergus (2014). "Visualizing and Understanding Convolutional Networks". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, pp. 818–833. ISBN: 978-3-319-10590-1.

Zhang, Yu-Dong et al. (2019). "Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation". In: *Multimedia Tools and Applications* 78.3, pp. 3613–3632. ISSN: 1380-7501. DOI: 10.1007/s11042-017-5243-3.

Zhang, Hongyi, Yann Dauphin, and Tengyu Ma (2019). "Fixup Initialization: Residual Learning Without Normalization". In: *ArXiv* abs/1901.09321. URL: https://arxiv.org/abs/1901.09321v2.

Zhang, Hua et al. (2016). "SketchNet: Sketch Classification With Web Images". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ed. by IEEE.

Zhang, Liheng, Marzieh Edraki, and Guo-Jun Qi (2018). "CapProNet: Deep Feature Learning via Orthogonal Projections onto Capsule Subspaces". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc, pp. 5814–5823. URL: http://papers.nips.cc/paper/7823-cappronet-deep-feature-learning-via-orthogonal-projections-onto-capsule-subspaces.pdf.

Zhang, Yuting, Kibok Lee, and Honglak Lee (2016). "Augmenting Supervised Neural Networks with Unsupervised Objectives for Large-scale Image Classification". In: *PMLR 48:612-621*. URL: http://arxiv.org/pdf/1606.06582v1.

Zhang, Zhi et al. (2018). "Progressive Neural Networks for Image Classification". In: *CoRR* abs/1804.09803.

Zhao, Junbo Jake et al. (2015). "Stacked What-Where Auto-encoders". In: *CoRR* abs/1506.02351. URL: https://arxiv.org/abs/1506.02351v8.

Zhao, Xin, Xianheng Wang, and Hongkai Wang (2018). "Classification of Benign and Malignant Breast Mass in Digital Mammograms with Convolutional Neural Networks". In: *ISICDM 2018*. Ed. by ACM. ICPS. New York, New York: The Association for Computing Machinery, pp. 47–50. ISBN: 9781450365338. DOI: 10.1145/3285996.3286006.

Zhong, Zhun et al. (2017). "Random Erasing Data Augmentation". In: *ArXiv* abs/1708.04896.

Zhou, Yiren, Sibo Song, and Ngai-Man Cheung (2017). "On Classification of Distorted Images with Deep Convolutional Neural Networks". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Ed. by IEEE. IEEE, pp. 1213–1217. ISBN: 978-1-5090-4117-6. DOI: 10.1109/ICASSP.2017.7952349.

Zhu, Yi and Shawn Newsam (2015). "Land use classification using convolutional neural networks applied to ground-level images". In: *23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2015)*. Ed. by Mohamed Ali. New York, NY: The Association for Computing Machinery, Inc, pp. 1–4. ISBN: 9781450339674. DOI: 10.1145/2820783.2820851.

Zhu, Zhe et al. (2016). "Traffic-Sign Detection and Classification in the Wild". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ed. by IEEE.

Zhuang, Bohan et al. (2018). "Structured Binary Neural Networks for Accurate Image Classification and Semantic Segmentation". In: *CoRR* abs/1811.10413. URL: http://arxiv.org/pdf/1811.10413v2.

# A Appendix

## A.1 Concept Matrix

The Concept Matrix A.1 maps each paper to concepts of neural networks they support. These concepts are CNNs (Conv.), residual networks (Res.), inception networks (Inc.), capsule networks (Caps.), dense convolutional neural networks (Dense), and depthwise separable convolutional neural networks (Sep. Conv.).

Table A.1: Concept Matrix

| **Author** | **Conv.** | **Res.** | **Inc.** | **Caps.** | **Dense** | **Sep. Conv.** |
|---|---|---|---|---|---|---|
| Law, Y. Shen, and Seresinhe 2017 | x | | | | | |
| AlAfandy et al. 2019 | x | | | | | |
| B. Li et al. 2018 | x | | | | | |
| X. Zhao, Xianheng Wang, and H. Wang 2018 | x | | | | | |
| Nejad et al. 2017 | x | | | | | |
| Xuan Chen et al. 2018 | | x | x | | | |
| Lian et al. 2018 | x | x | | | | |
| Hori, Okada, and Nitta 2016 | x | | | | | |
| Slavkovikj et al. 2015 | x | | | | | |
| Shima and Omori 2018 | x | | | | | |
| Y. Zhu and Newsam 2015 | x | | | | | |
| Dao-Duc, Xiaohui, and Morère 2015 | x | | | | | |
| Tyas et al. 2017 | x | | | | | |
| Too et al. 2019 | x | | | | | |
| B. Wang et al. 2018 | x | | | | | |
| Sultana, Sufian, and Dutta 2019 | x | x | x | x | | |

Table A.1: Concept Matrix

| Author | Conv. | Res. | Inc. | Caps. | Dense | Sep. Conv. |
|---|---|---|---|---|---|---|
| W. Chen et al. 2019 | | x | | | | |
| Tensmeyer and Martinez 2017 | x | | | | | |
| Xiong, H. J. Kim, and Hedau 2019 | x | | | | | |
| Madrazo et al. 2019 | x | | | | | |
| Shetty 2016 | x | | | | | |
| T. He et al. 2018 | | x | x | x | x | |
| Alom et al. 2018 | | | x | | | |
| Bejnordi et al. 2017 | | x | | | | |
| Vaila, Chiasson, and Saxena 2019 | x | | | | | |
| Stamoulis et al. 2018 | x | | | | | |
| McDonnell and Vladusich 2015 | x | | | | | |
| Basha et al. 2020 | x | | | | | |
| Cirean, Meier, and Schmidhuber 2012 | x | | | | | |
| Le Hou et al. 2015 | x | | | | | |
| Mendes and Silva 2018 | | x | | | | |
| Zhuang et al. 2018 | x | | | | | |
| Ju, Bibaut, and van der Laan 2018 | x | | x | | | |
| Hua Zhang et al. 2016 | x | | | | | |
| Jiménez-Sánchez, Albarqouni, and Mateus 2018 | x | | | x | | |
| Afshar, Plataniotis, and Mohammadi 2018 | | | | x | | |

Table A.1: Concept Matrix

| Author | Conv. | Res. | Inc. | Caps. | Dense | Sep. Conv. |
|---|---|---|---|---|---|---|
| Debayle, Hatami, and Gavet 2018 | x | | | | | |
| Iesmantas and Alzbutas 2018 | | | | x | | |
| Kieffer et al. 2017 | x | | | | | |
| Q. Liu et al. 2020 | x | | | | | |
| Das et al. 2018 | x | | | | | |
| Pimkin et al. 2018 | | x | | | x | |
| Y. Sun et al. 2019 | x | x | x | | x | |
| Z. Gao et al. 2017 | x | | | | | |
| Leroux et al. 2018 | x | | | | | |
| Matsunaga et al. 2017 | | x | | | | |
| Castelluccio et al. 2015 | x | | x | | | |
| Blot, Cord, and Thome 2016 | x | | | | | |
| Zhou, Song, and Cheung 2017 | x | | | | | |
| Z. Zhang et al. 2018 | x | | | | | |
| X. Shen et al. 2016 | x | | | | | |
| Nazeri, Aminpour, and Ebrahimi 2018 | x | | | | | |
| Yanhui Guo et al. 2018 | x | | x | | | |
| Howard 2013 | x | | | | | |
| Vo, Tran, and T. H. Le 2017 | x | | | | | |
| Roth et al. 2015 | x | | | | | |
| L. Xu et al. 2019 | | | | | x | |
| Fuyong et al. 2018 | x | | | | | |
| Chagas et al. 2018 | x | x | x | | | |

Table A.1: Concept Matrix

| Author | Conv. | Res. | Inc. | Caps. | Dense | Sep. Conv. |
|---|---|---|---|---|---|---|
| Shi et al. 2019 | x | | x | | | |
| Peng et al. 2018 | | | x | | | |
| Seo and Shin 2018 | | | x | | | |
| Kesim, Dokur, and Olmez 2019 | x | | | | | |
| Byerly, Kalganova, and Dear 2020 | | | | x | | |
| C. Xie et al. 2019 | | | | | | x |
| Sato, Nishimura, and Yokoi 2015 | x | | | | | |
| Cubuk et al. 2018 | | x | | | | x |
| Izmailov et al. 2018 | | x | | | | |
| Chang and Y.-S. Chen 2015 | x | | | | | |
| Darlow et al. 2018 | x | x | x | | x | |
| Yun et al. 2019 | | x | | | | |
| Chang and Y.-S. Chen 2017 | | x | | | | |
| G. Huang et al. 2017 | | | | | x | |
| Liang, Khoo, and H. Yang 2018 | | x | x | | x | |
| Tan and Le V 2019 | | | | | | x |
| Xiao Wang et al. 2019 | | x | | | | |
| Dong et al. 2017 | | x | x | | | |
| D. Mahajan et al. 2018 | | x | x | | | |
| S. Lim et al. 2019 | | x | x | | | x |
| Touvron et al. 2019 | | x | x | | | x |
| Hongyi Zhang, Dauphin, and Ma 2019 | | x | x | | | |

Table A.1: Concept Matrix

| Author | Conv. | Res. | Inc. | Caps. | Dense | Sep. Conv. |
|---|---|---|---|---|---|---|
| Y. Huang et al. 2019 | | | | | | x |
| Devries and Taylor 2017 | | x | | | | |
| Ji, Vedaldi, and Henriques 2019 | | x | | | | |
| Kolesnikov et al. 2019 | | x | | | | |
| HasanPour et al. 2016 | x | | | | | |
| Gong et al. 2020 | | | | | | x |
| Berthelot et al. 2019 | | x | | | | |
| Cai, L. Zhu, and Han 2018 | | x | | | | |
| Zhong et al. 2017 | | x | x | | | |
| Wan et al. 2013 | x | | | | | |
| S. Gao et al. 2019 | | x | x | | | |
| Kowsari et al. 2018 | x | | | | | |
| Sosnovik, Szmaja, and Smeulders 2020 | x | | | | | |
| Q. Xie et al. 2019 | | | | | | x |
| Denton, Gross, and Fergus 2016 | x | | | | | |
| Hu, L. Shen, and G. Sun 2018 | | x | | | | |
| J. J. Zhao et al. 2015 | x | | | | | |
| Assiri 2019 | x | | | | | |
| Jayasundara et al. 2019 | | | | x | | |
| Nøkland and Eidnes 2019 | x | | | | | |
| Harris et al. 2020 | | x | | | x | |
| Zagoruyko and Komodakis 2016 | | x | | | | |
| Rawat and Z. Wang 2017 | | x | x | | | |

Table A.1: Concept Matrix

| Author | Conv. | Res. | Inc. | Caps. | Dense | Sep. Conv. |
|---|---|---|---|---|---|---|
| Orhan 2018 | x | | | | | |
| Kubilius et al. 2019 | x | | | | | |
| L. Zhang, Edraki, and Qi 2018 | | | | x | | |
| Sabour, Frosst, and G. E. Hinton 2017 | | | | x | | |
| Lenssen, Fey, and Libuschewski 2018 | | | | x | | |
| Krizhevsky, Sutskever, and G. E. Hinton 2012 | x | | | | | |
| R. J. Wang, X. Li, and Ling 2018 | x | | | | | |
| Siyuan Huang et al. 2019 | x | | | | | |
| Hahn, Pyeon, and G. Kim 2019 | | | | x | | |
| Kosiorek et al. 2019 | | | | x | | |
| Ahmed and Torresani 2019 | | | | x | | |
| Lu et al. 2019 | x | | | | | |
| C. Chen et al. 2019 | x | | | | | |
| Babaie et al. 2017 | x | | | | | |
| Kyrkou and Theocharides 2019 | x | | | | | |
| Z. Sun, Ozay, and Okatani 2016 | x | | | | | |
| Ponti et al. 2017 | x | x | x | | | |
| Belilovsky, Eickenberg, and Oyallon 2019 | x | | | | | |
| Garud et al. 2017 | | | x | | | |
| Zahavy et al. 2018 | x | | | | | |
| Su et al. 2018 | x | | x | | | |

Table A.1: Concept Matrix

| Author | Conv. | Res. | Inc. | Caps. | Dense | Sep. Conv. |
|---|---|---|---|---|---|---|
| J. Yang, She, and M. Sun 2017 | x | x | | | | |
| Yan Wang et al. 2018 | x | | | | | |
| Y. Xie and Richmond 2019 | | | x | | | |
| Gurnani et al. 2019 | x | | | | | |
| Perez, Avila, and Valle 2019 | x | x | x | | x | |
| K. He et al. 2015b | x | | | | | |
| Beluch et al. 2018 | x | | | | | |
| Ioannou et al. 2015 | x | | | | | |
| J. Liu et al. 2016 | x | | | | | |
| P.-K. Kim and K.-T. Lim 2017 | x | | | | | |
| Fadaeddini, Eshghi, and Majidi 2018 | | x | x | | | |
| Levi and Hassncer 2015 | x | | | | | |
| Teramoto et al. 2017 | x | | | | | |
| Gallego, Pertusa, and Gil 2018 | x | x | x | | | x |
| Spanhol et al. 2016 | x | | | | | |
| Araújo et al. 2017 | x | | | | | |
| Sladojevic et al. 2016 | x | | | | | |
| Esteva et al. 2017 | x | | | | | |
| Ciresan et al. 2011 | x | | | | | |
| Y.-D. Zhang et al. 2019 | x | | | | | |
| Krizhevsky, Sutskever, and G. E. Hinton 2017 | x | | | | | |
| Al-Saffar, Tao, and Talab 2017 | x | | | | | |

Table A.1: Concept Matrix

| Author | Conv. | Res. | Inc. | Caps. | Dense | Sep. Conv. |
|---|---|---|---|---|---|---|
| Cheng et al. 2016 | x | | x | | | |
| Bentes, Velotto, and Tings 2018 | x | | | | | |
| T. Guo et al. 2017 | x | | | | | |
| Harangi 2018 | x | | | | | |
| Nogueira, Penatti, and dos Santos 2017 | x | | x | | | |
| Elhoseiny, Sheng Huang, and Elgammal 2015 | x | | | | | |
| Rastegari et al. 2016 | x | | | | | |
| Jeon and J. Kim 2017 | x | | | | | |
| H. Xu et al. 2019 | x | | | | | |
| Tokozume, Ushiku, and Harada 2018 | x | x | x | | x | |
| Taha et al. 2020 | | x | x | | x | |
| Xiaosong Wang et al. 2017 | x | x | x | | | |
| Kortylewski et al. 2020 | x | | | | | |
| Murthy et al. 2016 | x | | | | | |
| K. He et al. 2015a | x | | x | | | |
| Dvornik, Schmid, and Mairal 2019 | | x | | | | |
| Yiru Wang et al. 2019 | x | | | | | |
| Wu et al. 2015 | x | | | | | |
| Phan et al. 2020 | x | | | | | |
| F. Wang et al. 2017 | | x | | | | |
| B. Singh et al. 2018 | x | | | | | |
| H.-M. Yang et al. 2018 | x | | | | | |
| Gowal et al. 2019 | x | | | | | |

Table A.1: Concept Matrix

| Author | Conv. | Res. | Inc. | Caps. | Dense | Sep. Conv. |
|---|---|---|---|---|---|---|
| Y. Zhang, K. Lee, and H. Lee 2016 | x | | x | | | |
| B. Liu et al. 2015 | x | | | | | |
| T. Xiao et al. 2015 | x | | | | | |
| van Horn et al. 2018 | | x | x | | | |
| Z. Zhu et al. 2016 | x | | | | | |
| Z. Chen et al. 2020 | x | x | | | | |
| K. He et al. 2016 | | x | | | | |
| Szegedy, W. Liu, et al. 2015 | | | x | | | |
| Lecun et al. 1998 | x | | | | | |
| Szegedy, Ioffe, et al. 2017 | | x | x | | | |
| Lin, Q. Chen, and S. Yan 2013 | x | | | | | |
| C. Liu et al. 2018 | | | | | | x |
| Real et al. 2019 | | | | | | x |
| Yamada et al. 2019 | | x | | | | |
| Gastaldi 2017 | | x | | | | |
| Simonyan and Andrew Zisserman 2014 | x | | | | | |
| Zeiler and Fergus 2014 | x | | | | | |
| Chollet 2017 | | x | x | | | x |

## A.2 Benchmark Datasets

Performance of neural networks for image classification is measured on benchmark datasets. The benchmark datasets are listed below.

- ImageNet[189]
- CIFAR-10[190]
- CIFAR-100[191]
- MNIST[192]
- SVHN[193]
- STL-10[194]
- DeepFashion[195]
- FashionMNIST[196]
- CINIC-10[197]
- Flowers-102[198]
- Food-101[199]
- iNaturalist[200]
- Stanford Cars[201]
- EMNIST Letters[202]
- Kuzushiji-MNIST[203]
- CUB-200-2011[204]
- MulitMNIST[205]

[189]Deng et al. 2009.
[190]Krizhevsky 2012.
[191]Krizhevsky 2012.
[192]LeCun, Cortes, and Burges 2010.
[193]Netzer et al. 2011.
[194]Coates, Ng, and H. Lee 2011.
[195]Z. Liu et al. 2016.
[196]H. Xiao, Rasul, and Vollgraf 2017.
[197]Darlow et al. 2018.
[198]Nilsback and A. Zisserman 2008.
[199]Bossard, Guillaumin, and Van Gool 2014.
[200]van Horn et al. 2018.
[201]Krause et al. 2013.
[202]Cohen et al. 2017.
[203]Clanuwat et al. 2018.
[204]Wah et al. 2011.
[205]Sabour, Frosst, and G. E. Hinton 2017.

- ISIC[206]

## A.3 Inclusion and Exclusion Criteria

The inclusion and exclusion criteria determine whether or not a paper is selected for the literature review. The inclusion and exclusion criteria are listed below.

- Inclusion criteria
  - Image classification task
  - Realized with any neural network
  - Included in a journal, conference proceeding, peer reviewed or at least cited by such papers
- Exclusion criteria
  - Other tasks, for example, semantic segmentation, object detection, image generation, etc.
  - Realized with models which are not neural networks
  - If the paper was refuted or updated by a more recent paper

## A.4 Literature Source List

The literature source list lists all sources of papers. These sources are journals, conference proceedings, full-text databases, image classification leaderboards, and scientific search engines. The literature source list is displayed below.

- Journals and conference proceedings
  - Proceedings of Machine Learning Research [207]
  - Neural Information Processing Systems [208]
  - The Computer Vision Foundation [209]
  - The Institute of Engineering Technology Image Processing [210]

[206] Combalia et al. 2019; Codella et al. 2018; Tschandl, Rosendahl, and Kittler 2018.
[207] `http://proceedings.mlr.press/`
[208] `https://papers.nips.cc/`
[209] `https://openaccess.thecvf.com/menu.py`
[210] `https://digital-library.theiet.org/content/journals/iet-ipr`

  - The Institute of Engineering Technology Computer Vision [211]
  - International Journal of Computervision [212]
  - European Conference on Computer Vision [213]
  - International Conference on Computer Vision [214]
- Full-text databases
  - ArXiv [215]
  - Open Review [216]
  - Association for Computing Machinery Digital Library [217]
  - Institute of Electrical and Electronics Engineers Xplore [218]
  - Semanticscholar [219]
  - Massachusetts Institute of Technology [220]
  - Elsevier Sciencedirect [221]
  - De Gruyter [222]
  - JSTOR [223]
- Image Classification Leaderboards
  - Image Classification Leaderboards Papers with Code [224]
  - Computer Vision Leaderboard [225]
  - International Skin Imaging Collaboration [226]
- Bibliographies and scientific search engines

[211] https://digital-library.theiet.org/content/journals/iet-cvi
[212] https://www.springer.com/journal/11263
[213] https:link.springer.com/conference/eccv
[214] https:icmv.org/pro.html
[215] https://arxiv.org
[216] https://openreview.net
[217] https://dl.acm.org
[218] https:ieeexplore.ieee.org
[219] https://semanticscholar.org
[220] https://mitpressjournals.org
[221] https://sciencedirect.com
[222] https://degruyter.com
[223] https://jstor.org
[224] https://paperswithcode.com/task/image-classification
[225] github.com/kobiso/Computer-Vision-Leaderboard
[226] https://challenge2019.isic-archive.com/leaderboard.html

- Library catalogue of the Duale Hochschule Baden-Württemberg Mannheim
- Digital Bibliography & Library Project [227]
- Google Scholar [228]

## A.5 Term Table

The term table lists terms related to the problem statement described in Section 1.2. As a result, the term table contains all terms that can be used to derive search queries. The term table is displayed below.

Table A.2: Term Table

| **Term** | **General Term** | **Subsumable Term** | **Synonym** | **Related Term** |
|---|---|---|---|---|
| Image Classification | Computer Vision, Classification, Image Processing, Image Analysis | Multi-Class Classification, Single-Label Classification, Multi-Label Classification, Emotion Classification, Person-Re-Identification | Image Categorization, Image Label Prediction | Semantic Segmentation, Object Detection, Image Generation |

[227] https://dblp.org
[228] https://scholar.google.com

Table A.2: Term Table

| **Term** | **General Term** | **Subsumable Term** | **Synonym** | **Related Term** |
|---|---|---|---|---|
| Neural Network | Computer Science, Data Sciene, Artificial Intelligence, Computational Graph, Machine Learning | MLP, FNN, RNN, CNN, CNN-LSTM, GRU, GAN, Tree RNN, Attention, Transformer, HighwayNets, ResNets, CRNN, RCNN, mLSTM, mRNN, DenseNets, CapsNets, Inception, DNN, ANN | | |
| Tool | | | implement, apperatus, gimmick, instrument, gear, equipment, utensil, gadget | |
| Dataset | | ImageNet, CIFAR, MNIST, SVHN, STL Clothing, CINIC, Flowers, iNaturalist, Stanford Cars, CUB | | |
| SOTA | | | | Review, Survey, Benchmark |

## A.6 Contents of the Digital Appendix

The dataset and code used to conduct the experiment are appended in the digital appendix. The digital appendix is provided on the enclosed USB flash drive *D*. The contents of the digital appendix are listed in Table A.3.

Table A.3: Digital Appendix

| **Name** | **Path** | **Description** |
|---|---|---|
| TIC Dataset | `D/dataset` | This folder contains the images of the TIC Dataset and its license. The images are structured in six folders. Each folder contains the images of one class. The folder is named after the class. |
| Dataset Splitter | `D/split_dataset.py` | This file contains python code splitting the dataset into a training, validation, and test dataset. |
| Dataset Pre-Loader | `D/preload_dataset.py` | This file contains python code loading and storing the training, validation, and test dataset as arrays. For each of those splits, the images are loaded, resized to 224 by 224 pixels, and formatted as arrays. The resulting image arrays are stored in array X. For each image array, a class array is created. The class array is stored in array Y. Image array and corresponding class array share the same index. In consequence, six arrays are created, two for each split. These arrays are X Train, Y Train, X Val, Y Val, X Test, and Y Test. |

Table A.3: Digital Appendix

| Name | Path | Description |
|---|---|---|
| X Train | `D/x_train.npy` | This file contains an array of resized image arrays of the training dataset. The images are resized to 224 by 224 pixels. |
| Y Train | `D/y_train.npy` | This file contains an array of class arrays of the training dataset. |
| X Val | `D/x_val.npy` | This file contains an array of resized image arrays of the validation dataset. The images are resized to 224 by 224 pixels. |
| Y Val | `D/y_val.npy` | This file contains an array of class arrays of the validation dataset. |
| X Test | `D/x_test.npy` | This file contains an array of resized image arrays of the test dataset. The images are resized to 224 by 224 pixels. |
| Y Test | `D/y_test.npy` | This file contains an array of class arrays of the test dataset. |
| VGG-19 Code | `D/vgg.py` | This file contains python code implementing VGG-19. |
| ResNet-152 Code | `D/resnet.py` | This file contains python code implementing ResNet-152. |
| ResNeXt-101 Code | `D/resnext.py` | This file contains python code implementing ResNeXt-101. |
| DenseNet-264 Code | `D/densenet.py` | This file contains python code implementing DenseNet-264. |

Table A.3: Digital Appendix

| Name | Path | Description |
|---|---|---|
| EfficientNet-B7 Code | `D/efficientnet.py` | This file contains python code implementing EfficientNet-B7. |
| Train VGG-19 | `D/train_vgg.py` | This file contains python code training and evaluating VGG-19. |
| Train ResNet-152 | `D/train_resnet.py` | This file contains python code training and evaluating ResNet-152. |
| Train ResNeXt-101 | `D/train_resnext.py` | This file contains python code training and evaluating ResNeXt-101. |
| Train DenseNet-264 | `D/train_densenet.py` | This file contains python code training and evaluating DenseNet-264. |
| Train EfficientNet-B7 | `D/train_efficientnet.py` | This file contains python code training and evaluating EfficientNet-B7. |

# Declaration of Authorship

I hereby declare that the paper submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. Furthermore, I assure that the electronic version submitted corresponds to the printed version.

Place / Date Fabian Wolf