


Article

A Novel Architecture to Classify Histopathology Images Using Convolutional Neural Networks

Ibrahim Kandel * and Mauro Castelli 

Nova Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisbon, Portugal; mcastelli@novaims.unl.pt

* Correspondence: D20181143@novaims.unl.pt

Received: 24 February 2020; Accepted: 20 April 2020; Published: 23 April 2020



Abstract: Histopathology is the study of tissue structure under the microscope to determine if the cells are normal or abnormal. Histopathology is a very important exam that is used to determine the patients' treatment plan. The classification of histopathology images is very difficult to even an experienced pathologist, and a second opinion is often needed. Convolutional neural network (CNN), a particular type of deep learning architecture, obtained outstanding results in computer vision tasks like image classification. In this paper, we propose a novel CNN architecture to classify histopathology images. The proposed model consists of 15 convolution layers and two fully connected layers. A comparison between different activation functions was performed to detect the most efficient one, taking into account two different optimizers. To train and evaluate the proposed model, the publicly available PatchCamelyon dataset was used. The dataset consists of 220,000 annotated images for training and 57,000 unannotated images for testing. The proposed model achieved higher performance compared to the state-of-the-art architectures with an AUC of 95.46%.

Keywords: histopathology images; deep learning; convolutional neural networks; image classification

1. Introduction

Cancer, also called malignancy and neoplasms, is an abnormal growth of cells in a multistage process that generally progresses from a pre-cancerous lesion to a malignant tumor, which can then invade adjoining parts of the body and spread to other organs. It is considered the second leading cause of death globally, and it is responsible for an estimated 9.6 million deaths in 2018 [1,2]. There are more than 100 types of cancer, including breast cancer, skin cancer, lung cancer, colon cancer, prostate cancer, and lymphoma [1]. Through the blood and the lymph systems of the body, cancer can spread across the entire body affecting different organs. It is perceived to be the result of the interaction between a person's genetic factors and exposure to certain environmental external agents, known as carcinogens. Usually, the definitive cancer diagnosis is through histopathology, the study of the tissue structure. Histopathology is done by studying the tissues under a microscope to detect any cell alterations. It is one of the most important steps in the treatment plan because it can help in the early detection of cancer [3]. Histopathology study is a very time-consuming process and requires a very experienced pathologist. Due to its difficulty, a second opinion from another pathologist is often needed, especially for certain kinds of tumors. Even very experienced pathologists can disagree with each other in the classification of the histopathology images [4]. Moreover, the number of active pathologists decreased dramatically in the last decade. For example, in the US, the number of pathologists decreased by 17.5%, which led to an increase in the workload by more than 40% [5]. Currently, the used microscopes are digital, meaning that they can produce a digital image that can be viewed and stored on the computers. These digital images can be used to make an automatic classifier. The automatic classification of histopathology will save a lot of time and can give a second opinion to the pathologists.

Deep learning architectures have demonstrated their suitability to successfully address optimization problems over different domains. A convolutional neural network (CNN) corresponds to a particular type of deep learning model, and it was proposed by LeCun et al. [6] to address problems in the computer vision domain. The real advancement happened in 2012 when Krizhevsky et al. [7] won the ILSVRC challenge [8] with an accuracy of 84.6%. Since then, CNNs are considered state-of-the-art models for image classification. CNNs have been successfully applied in different fields like traffic sign classification [9,10], text classification [11,12], speech recognition [13,14], and machine translation [15,16].

In this work, we present a novel CNN architecture to classify lymph node stained histopathology images. The publicly available PatchCamelyon dataset [17,18] was used to train and test the proposed architecture. The main contributions of this paper are as follows:

- We propose a novel CNN architecture that can classify histopathology images with high accuracy.
- We investigate the impact of dropout layers and the impact of the location of the normalization layer.
- We test six activation functions to study their impact on the proposed architecture, rather than choosing the de-facto ReLU activation function.
- We study the impact of two different optimizers on the CNN performance.
- We consider four popular state-of-the-art CNNs to compare the performance of our model. These CNNs were trained on the PatchCamelyon dataset. These results can be used by researchers instead of training these models again from scratch, which can take hours (if not days), especially in the lack of computational power.

The rest of this paper is organized as follows: In Section 2, we review the previous studies that introduced novel CNN architectures. The proposed methodology is stated in Section 3. The obtained results are shown in Section 4. The discussion and conclusion are presented in Sections 5 and 6, respectively.

2. Literature Review

In recent years, many algorithms were introduced to help in classifying histopathology images. Nowadays, CNNs are considered as the state-of-the-art algorithm for classifying images. In 1989, Lecun et al. [19] presented the first CNN with 5 convolution layers. Before Lecun's work, CNN was not a popular choice for image classification because of the computational cost and the (small) size of the available datasets. In 2012, Krizhevsky et al. [7] re-introduced CNNs by winning the ImageNet challenge with their AlexNet CNN that obtained a 16% classification error rate compared to 25% of the second-place model. Since then, CNN became the de-facto algorithm for image classification, and many CNNs were subsequently defined.

To address an image classification task by using CNNs, it is possible to rely on two different approaches: use a pre-existing architecture that was developed to classify natural images or, as done in this paper, develop a novel architecture. This section focuses on novel architectures that were introduced mainly to classify histopathology images. We summarize the recent contributions in the area of histopathology images in Table 1, while Table 2 reports the important information associated with the architectures presented in the papers of Table 1. For a complete description of the architectures considered in Tables 1 and 2, the reader is referred to their respective paper.

Table 1. Recent studies in the area of histopathological image classification.

Authors	Main Contribution	Number of Classes	Metric	Test Set Performance
Nguyen et al. [20]	<ul style="list-style-type: none"> A CNN is used to classify breast cancer images belonging to 8 classes (four benign subclasses and four malignant subclasses). This was the first attempt to classify breast cancer images in eight subclasses. 	8	Accuracy	73.68%
Bayramoglu et al. [21]	<ul style="list-style-type: none"> A single task CNN is used to predict malignancy from breast cancer images. A multi-task CNN is used to predict both malignancy and image magnification level simultaneously (for a total of eight classes). Efficient method that allows to use new data with the same or different magnification levels than previous data. 	2 for the malignant/ not malignant task 8 for the magnification task	Accuracy	83.25% 80.10%
Arjmand et al. [22]	<ul style="list-style-type: none"> Fully automated diagnostic tool for non-alcoholic fatty liver disease classification, based on an optimized CNN architecture. 	2	Accuracy	95%
Sirinukunwattana et al. [23]	<ul style="list-style-type: none"> Deep learning model for nucleus detection and classification from histology images of colorectal adenocarcinomas. Novel Neighbouring Ensemble Predictor (NEP) coupled with CNN to more accurately predict the class label of detected cell nuclei. 	4	F1 Score	0.692 (This is the combined performance on nucleus detection and classification)
Lai et al. [24]	<ul style="list-style-type: none"> Deep learning model that integrates Coding Network with Multilayer Perceptron (CNMP). Combination of high-level features that are extracted from a deep convolutional neural network with traditional features of an image that can be extracted using simple image analysis concepts. 	2	Accuracy	90.1% and 90.2% on two benchmark medical image datasets
Basha et al. [25]	<ul style="list-style-type: none"> Definition of a CNN architecture for the classification of histological routine colon cancer nuclei. Significant reduction of the number of learnable parameters compared to the popular CNN models such as AlexNet, and GoogLeNet. 	4	F1 Score	0.7887

Table 2. Information related to the architectures defined in the papers identified in Table 1.

	Conv Layers	FC layers	Dropout Layers	Normalization Layers	Activation Function	Pooling Layers
Nguyen et al. [20]	5	1	3	6	LeakyReLU	3
Bayramoglu et al. [21]	3	2	2	2	ReLU	3
Arjmand et al. [22]	3	1	2	3	ReLU	2
Sirinukunwattana et al. [23]	2	2	0	0	ReLU	2
Lai et al. [24]	6	0	0	0	ReLU	2
Basha et al. [25]	4	2	2	6	ReLU	2

To assess the performance of the model proposed in this paper, we perform a comparison against the performance of three state-of-the-art models, namely, VGG, InceptionV3, and ResNet architectures. The following paragraphs provide a short description of these models. For a complete description, the reader is referred to the papers where these models were originally presented.

2.1. VGG Architectures

Simonyan et al. [26] proposed a novel CNN called VGG, which achieved an 8.1% error rate, a great achievement compared to the AlexNet network. In particular, two main architectures were introduced: VGG16 and VGG19. The main difference among them is in the number of convolution layers. VGG16 consisted of 13 convolution layers and 3 fully connected layers, while VGG19 considered 16 convolution layers and 3 fully connected layers. All the convolution layers have a 3×3 kernel size, with the number of kernels ranging from 64 till 512. VGG16 can be divided into 5 convolution blocks, where each block contains three convolution layers, followed by a max-pooling layer. VGG16 with fully connected layers has 138 million parameters, whereas VGG19 with fully connected layers has 144 million parameters.

2.2. InceptionV3 Architecture

InceptionV3 is a recent CNN architecture with 22 layers that was introduced by Szegedy et al. [27]. The main difference of this architecture to others is the fact that it connects the convolutions in parallel instead of connecting them sequentially. The authors named this module the inception module. The point of the inception module is to process the images at different scales. The InceptionV3 architecture consists of 9 inception modules and one fully connected layer. In total, InceptionV3 has 23.8 million parameters.

2.3. ResNet Architecture

He et al. [28] noticed that the CNN accuracy will get saturated as soon as the model gets 30 layers deep. The main reason for that saturation is the vanishing gradients problem, where the bottom layers of the CNN will stop being updated. The authors introduced ResNet architecture that could overcome the problem of vanishing gradients. The main difference between ResNet architecture and other networks is the residual connection, where this connection will skip a few convolution layers at a time. ResNet architecture won the ImageNet challenge in 2015 with an error rate of 3.57% that surpassed the human error rate for the first time. In total, ResNet has 25 million parameters.

3. Methodology

This section describes the proposed CNN and the dataset used in this study.

3.1. Proposed Architecture

The proposed CNN was chosen by analyzing the extant literature in the area, thus exploiting the contributions of different works. The main inspiration of the proposed model is the VGG16 architecture [26], with the main differences being (1) the use of normalization layer in every convolution block, (2) the use of three convolution layers in every block instead of four, and (3) the use of fully connected layers with 512 neurons instead of 4096. The proposed architecture is shown in Figure 1. In the proposed architecture, the input image size is 96×96 pixels. The architecture has 5 convolution blocks that act as feature extractors and one fully-connected block that acts as a classifier. In the first convolution block, three convolution layers followed by a batch normalization layer and a max-pooling layer are used. The padding is kept the same for the three convolution layers to make use of every pixel, especially in the first convolution block and, for the same reason, the stride is set to 1. For the three convolution layers, 32 kernels were used with a size of 3×3 . To reduce the dimensions by two to keep the most relevant features obtained from the first convolution block, a max-pooling layer is added. The second, third, and fourth convolution blocks use the same hyperparameters used in the first convolution block, except for the number of kernels used, which was set to 64, 128, and 256, respectively. In the fifth convolution block, the second and the third convolution layers have no padding to decrease the spatial information with 512 kernels. The output of the convolution blocks is flattened and, subsequently, it becomes the input of the classifier block. The classifier block has two fully connected layers, each with 512 neurons. Overall, the inputs to the convolution blocks are $96 \times 96 \times 3$, $48 \times 48 \times 32$, $24 \times 24 \times 64$, $12 \times 12 \times 128$, and $6 \times 6 \times 256$, respectively. The final layer is a sigmoid function that is used to classify into two classes. In total, the proposed architecture consists of 15 convolution layers, 2 fully connected layers, 5 pooling layers, and 5 normalization layers.

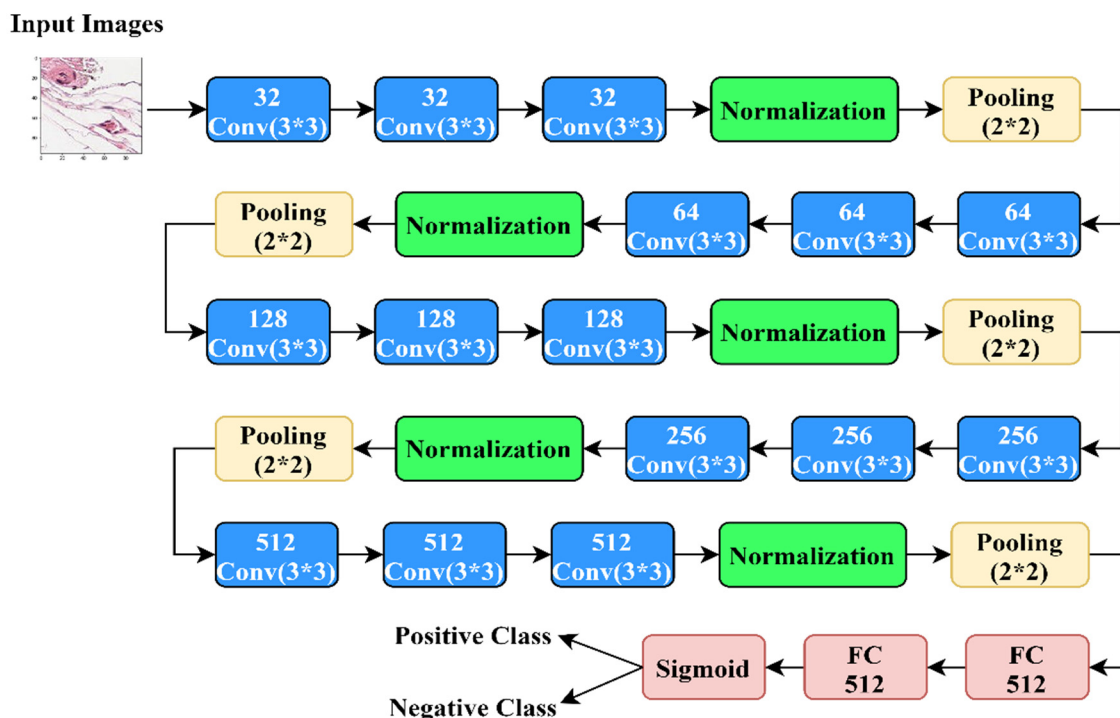


Figure 1. A schematic diagram of the proposed network architecture.

Activation functions have a huge impact on the speed and the accuracy of the networks, and that is why, instead of using the ReLU activation function which is considered as the de-facto activation function, five additional activation functions were tested to determine the best for the proposed architecture.

Four designs were tested to determine the optimal architecture. The first design is to test the effect of dropout layers in the classifier block, where a dropout layer with a dropout ratio of 50% will be inserted after each fully connected layer. The second design is to test the performance of the architecture without the dropout layers. The third design is to test the performance of the CNN after placing the normalization after the activation function. A dropout layer with a 50% dropout ratio will be added after each fully-connected layer. The fourth design is the same as the third design but without dropout layers. Different designs are shown in Figure 2.

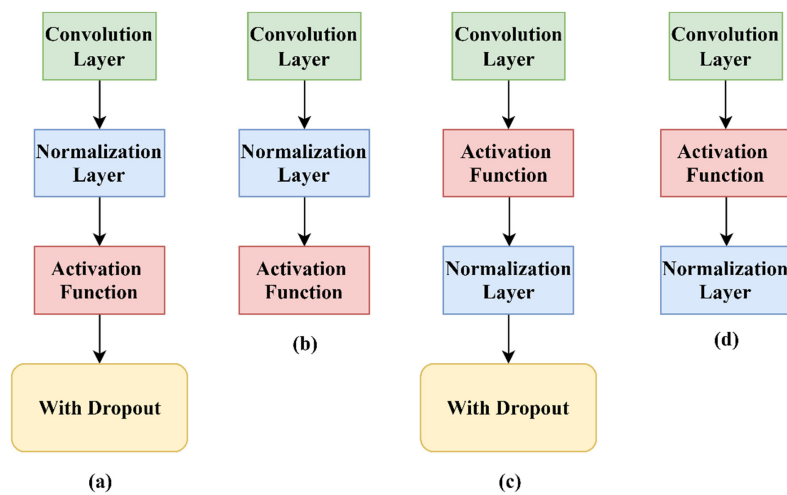


Figure 2. Different designs tested: (a) shows the first design, the proposed architecture with the normalization layer placed before the activation function, and two dropout layers are placed in the classifier block; (b) shows the second design, which is similar to the first design but without any dropout layers; (c) shows the third design, where the normalization layers are placed after the activation function, and two dropout layers are placed in the classifier block; and (d) shows the fourth design, which is similar to the third design but without any dropout layers.

3.2. Dataset

As described in Bejnordi et al. [18], the Camelyon dataset was sampled from 399 patients from two hospitals in the Netherlands. The dataset was annotated with the help of experienced pathologists from the Netherlands. In particular, the dataset labels were manually annotated by two students and, subsequently, were checked in deep detail by two experienced pathologists. To check the performance of the pathologists on this dataset, two sets of experiments were performed.

The first experiment was performed without any time constrain, and 11 experienced pathologists were asked to annotate a first subset of the images. In the second experiment, a two-hour time limit was given to the experts for annotating a second subset of images. The challenge organizers chose the AUC of the ROC curve as an evaluation criterion for this competition, thus every score is presented in terms of AUC of the ROC curve. The AUC score achieved by the first experiment was 96.6%, and the score of the second experiment was 81%.

The PatchCamelyon dataset [17,18] is an extension of the Camelyon dataset, which contains 277,000 histopathology images with an image size of 96×96 pixels at $10\times$ magnification. A total of 220,000 images are annotated with 60% positive cases and 40% negative cases. A total of 57,000 images are un-annotated images to test the classifier, and there are no duplicates in the PatchCamelyon dataset. The resulting model is subsequently used to predict the labels of the test set and finally uploaded to the Kaggle platform to obtain the model AUC. This process was necessary since the test labels are available only to the owner of the data. To increase the size of the dataset and to make the model more robust to overfitting, the following augmentation techniques were applied: horizontal flip, vertical flip, rotation range, zoom range, width shift range, height shift range, shear range, and channel shift range. Figure 3 shows two images of the considered dataset.

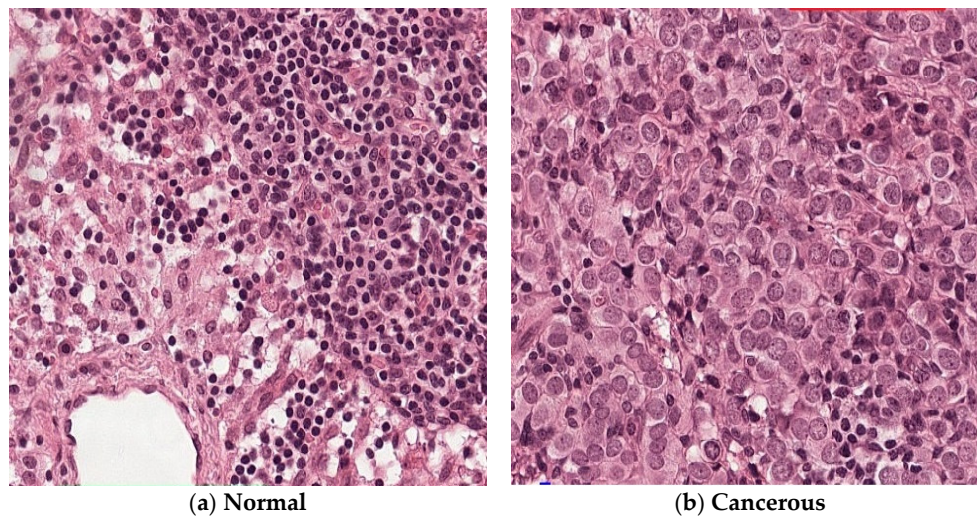


Figure 3. Samples of the PatchCamelyon dataset. (a) A normal sample. (b) A cancerous sample.

4. Results

This section presents the hyperparameters used in the experiments, the results obtained using different activation functions, the results of using different designs, and the results obtained using different state-of-the-art CNN architectures.

4.1. Experimental Setup

Six activation functions were used to determine the optimum for the proposed architecture. The tested activation functions are Tanh, Sigmoid, ReLU [29], LeakyReLU [30], ELU [31], and SELU [32]. Two optimizers were used as well, namely, Adam [33] and RMSProp [34]. The PatchCamelyon dataset was divided into 80%/20% for training and validation. For all the experiments, the batch size used was 64. Because of the size of the training dataset and the computational cost of CNN training, early stopping of 10 epochs was used for all the experiments. According to Shorten and Khoshgoftaar [35], image augmentation techniques can be used to increase the size of the dataset. This results in improved model performance and in the reduction of the overfitting that may occur when using small datasets. Thus, we applied image augmentation in all the experiments conducted in this paper, using the transformation specified in Section 3.2. The positive to negative ratio was kept the same after the application of the augmentation techniques.

All the experiments were implemented using Keras API [36] with TensorFlow API [37] in the backend.

4.2. Results

This section presents the results obtained and is divided into four parts. In the first part, we present the results obtained from training the proposed architecture using two optimizers and six different activation functions. In the second part, we present the results of the four different designs to enhance the performance of the network. In the third part, we present the results of the comparison of our architecture against state-of-the-art models. In the fourth part, we present the results of the comparison between our architecture and different architectures that were developed for histopathology image classification.

4.2.1. The Results of Different Activation Functions

Six different activation functions (and two different optimizers) were used to test the model performance. The sigmoid activation function achieved the lowest accuracy among all the activation functions taken into account over both the optimizers and so it was removed from other tests. The ReLU

activation function achieved a low accuracy compared to the other competitors, with comparable results obtained with the Adam and RMSProp optimizers. The LeakyReLU activation function scored the same as ReLU for the Adam optimizer but achieved a lower accuracy with the RMSprop optimizer. This could indicate that this activation function makes the network slightly unstable. The ELU activation function scored the best accuracy compared to all the functions, with comparable results for Adam and RMSProp. The SELU activation function had a high accuracy with Adam optimizer, while its performance degraded when using the RMSProp optimizer. The saturated function Tanh scored very high in both the optimizers, where the result of the RMSProp optimizer was higher than the Adam optimizer. Overall, the best achieving activation function was the ELU, followed by SELU and Tanh activation functions. The results are shown in Table 3.

Table 3. The AUC results of both optimizers for the different activation functions.

	ReLU	LeakyReLU	ELU	SELU	Sigmoid	Tanh
<i>Adam</i>	87.68%	87.37%	93.66%	92.73%	84.03%	91.70%
<i>RMSprop</i>	85.01%	83.02%	92.99%	88.43%	84.49%	92.00%

4.2.2. The Results of Different Designs

Four different designs were tested to check the performance of the network. The first design considers the network with two dropout layers in the classifier block, and the normalization layer before the activation layer. The second design is similar to the first design but without any dropout layers. The third design places the normalization layer after the activation layer and adds dropout layers in the classifier block. The fourth design is similar to the third design but without adding any dropout layers in the classifier block. Tables 4 and 5 show the results for both the Adam and RMSProp optimizers.

Table 4. The AUC results of the Adam optimizer.

	Tanh	ReLU	LeakyReLU	ELU	SELU
<i>H1</i>	91.70%	87.68%	87.37%	93.66%	92.73%
<i>H2</i>	92.96%	85.45%	87.17%	91.76%	92.82%
<i>H3</i>	94.39%	91.78%	89.96%	94.40%	90.16%
<i>H4</i>	95.46%	89.33%	89.11%	93.85%	92.71%

Table 5. The AUC results of the RMSProp optimizer.

	Tanh	ReLU	LeakyReLU	ELU	SELU
<i>H1</i>	92.00%	85.01%	83.02%	92.99%	88.43%
<i>H2</i>	94.09%	89.77%	90.37%	87.62%	94.25%
<i>H3</i>	93.35%	88.40%	90.38%	92.26%	93.86%
<i>H4</i>	93.43%	87.70%	90.11%	94.86%	91.58%

The ReLU activation function has the lowest score compared to other activation functions across the four designs. The best performance of ReLU (with Adam optimizer) was obtained in the third design, followed by the fourth design. The worst score was by using the second design. The results with the RMSprop optimizer were poorer than the Adam optimizer. The best result was achieved with the second design, followed by the third design. The lowest result was obtained with the first design. Overall, the best result achieved by ReLU was 91.78%. The LeakyReLU activation function had slightly lower accuracy than the ReLU function. With Adam optimizer, the third design was the best performer, followed by the fourth design. The worse result was obtained with the second design.

With the RMSprop optimizer, the best performance was obtained in the third design, followed by the second design. The worst performance was achieved by using the first design. The results of Adam and RMSprop optimizers were significantly different, especially for the first design. Overall, the best result achieved by LeakyReLU was 90.38%.

The ELU activation function had higher results compared to both ReLU and LeakyReLU. With Adam optimizer, the best performance was achieved by using the third design, followed by the fourth design. The lowest performance was achieved with the second design. With the RMSProp optimizer, the best result was achieved by using the fourth design, followed by the first design. The worst result was achieved with the second design. The results of both Adam and RMSprop optimizers were comparable, except for the second design. Overall, the best result achieved by the ELU activation function was 94.86%. The SELU activation function had slightly lower performance than the ELU activation function. With Adam optimizer, the best performance was achieved by using the second design, followed by the first design. The lowest performance was obtained with the third design. With the RMSprop optimizer, the best performance was achieved by using the second design, followed by the third design. The lowest performance was noticed with the first design. The results of Adam and RMSprop optimizers were significantly different, especially for the first design. Overall, the best result achieved by the SELU activation function was 94.25%.

The Tanh activation function achieved the highest results compared to all the other tested activation functions. With Adam optimizer, the best performance was obtained with the fourth design, followed by the third design. The lowest performance was achieved with the first design. With the RMSProp optimizer, the best performance was achieved by using the second design, followed by the fourth design. The lowest performance was obtained by using the first design. The results of both optimizers (Adam and RMSProp) were comparable. Overall, the best result achieved by the Tanh optimizer was 95.46%.

Using Adam optimizer, the first design was, overall, similar to the second design, meaning that the presence of the dropout layer did not increase the performance of the model. However, the performance of the third and fourth designs was higher, which indicates that the location of the normalization layer has an impact on the performance of the architecture. There are no significant differences between the third and the fourth designs, which indicates that the presence of the dropout layer does not increase the network performance.

Using the RMSProp optimizer, the performance of the first design was the lowest compared to the other designs. The second design achieved greater accuracy than the first design, which can indicate that the dropout layer can limit overfitting. The second, third, and fourth designs achieved a different performance, thus corroborating the hypothesis that the location of the normalization layer has a significant impact on the performance of the model. The third and the fourth designs were similar as well, indicating that the presence of the dropout layer has no effect on the model accuracy. All in all, based on the aforementioned results, we recommend practitioners to rely on the fourth design.

4.2.3. The Results over Benchmark CNN Architectures

In this section, the results of four benchmark CNN architectures are presented. Two sets of experiments were performed to compare our proposed architecture with four CNN popular benchmark architectures, namely, VGG16, VGG19, InceptionV3, and ResNet. The first set of experiments aims at comparing the architectures' performance under the first design (dropout layer in the classifier block). All the original classifier blocks of the CNN were removed and replaced by two fully connected layers with a dropout layer after each fully connected layer, with a dropout probability of 0.5. The second set of experiments compared the performance of the architectures under the fourth design (without a dropout layer in the classifier block). Just like the first set of experiments, all the original fully connected layers were removed and replaced with two fully connected layers without any dropout layers. All the architectures were trained from scratch (i.e., no transfer learning was used).

In the first sets of experiments, using the first design, two optimizers were used as well. By using the RMSprop optimizer, the best performing architecture was our proposed architecture, followed by the VGG19 network. The worse performance was achieved by the InceptionV3 architecture. By using Adam optimizer, the highest performance was obtained by our proposed architecture, followed by VGG19. The poorest performance resulted from the ResNet architecture. Overall, our proposed architecture outperformed the other architectures taken into account. The results of the first set of experiments are shown in Table 6. In the second set of experiments, using the fourth design, two optimizers were used as well. By using RMSprop, the highest performance was obtained by using our architecture, followed by VGG16. The lowest performance was achieved by using ResNet. By using Adam optimizer, the highest performance was achieved with our architecture, followed by VGG16. The lowest performance was obtained with the ResNet architecture. Overall, our proposed architecture achieved higher performance than the other tested CNNs. The results of the second set of experiments are shown in Table 7.

Table 6. The AUC results obtained with benchmark architectures under the first design.

	RMSProp	Adam
Our Model	92.99%	93.66%
VGG16	84.22%	89.53%
VGG19	89.08%	90.64%
InceptionV3	82.66%	82.47%
ResNet	85.24%	81.21%

Table 7. The AUC results obtained with benchmark architectures under the fourth design.

	RMSProp	Adam
Our Model	94.86%	95.46%
VGG16	89.33%	91.00%
VGG19	89.24%	89.20%
InceptionV3	87.15%	85.88%
ResNet	78.01%	83.52%

4.2.4. The Results of State-of-the-Art CNN Architectures

It is impossible to compare the results of our architecture against other architectures unless both the architectures were trained and tested on the same dataset and using the same hyperparameters like batch size, image augmentation, optimizer, and learning rate. That is why we trained different state-of-the-art CNN architectures on the PatchCamelyon dataset to easily compare our proposed CNN architecture with others. The image size was the only hyperparameter that was different between different architectures: The images were rescaled to follow the requirements of each architecture. Using the RMSProp optimizer, the best performance was achieved by using our proposed architecture followed by the architecture of Lai et al. [24]. The lowest performance was obtained with the Sirinukunwattana et al. [23] architecture. Using Adam optimizer, the best performance was achieved by our architecture, followed by Lai et al. [24] architecture. Similar results were obtained when the RMSProp optimizer was considered. Overall, our proposed architecture outperformed all the other architectures tested, as summarized in Table 8.

Table 8. The AUC results of the State-of-the-art architectures.

	RMSProp	Adam
Our Model	94.86%	95.46%
Bayramoglu et al. [21]	77.22%	86.68%
Arjmand et al. [22]	85.69%	88.23%
Lai et al. [24]	86.06%	92.11%
Sirinukunwattana et al. [23]	72.28%	68.85%
Basha et al. [25]	80.69%	75.04%
Nguyen et al. [20]	81.16%	86.68%

5. Discussion

In this work, a novel CNN architecture was proposed to classify histopathology images. This section discusses the results obtained.

5.1. Histopathology Images Importance And Challenges

Histopathology images classification is considered a very difficult task and very subjective as well, where two experienced pathologists can have very different opinions, and that is where an automatic classifier can be very important by providing a second opinion.

5.2. The Presented Architecture Choice

The architecture presented in this work was chosen after an extensive design phase, where different architectures were tested. As pointed out by Sirinukunwattana et al. [23], giving theoretical justification for the network architecture is very challenging and is still matter of ongoing research. Based on our results, three aspects affected the performance of the CNN: the position of the normalization layers in regards to the activation function, the presence of the dropout layers in the classifier block, and the activation function used.

5.3. The Effect of Different Activation Functions

Six activation functions were tested using our proposed architecture, two of them are saturated and four are non-saturated. The sigmoid function did not achieve a satisfying result and so it was removed from further testing. The Tanh function achieved outstanding results compared to other state-of-the-art non-saturated activation functions, which was quite surprising.

5.4. The Effect of the Location of the Normalization Layer and the Dropout Layer

Four different designs were tested to detect the optimal location of the normalization layer and the effect of the dropout layer on it. From the obtained results, we can conclude that the location of the normalization layer is very important, for which we recommend placing the normalization layer before the activation function. The presence of the dropout layer is not always guaranteed to increase network performance. Moreover, the optimizers played a very important role in the network performance, and from our results, Adam achieved better performance than RMSProp.

5.5. Comparison between Different Benchmark CNN

From our experiments, we noticed that adding a dropout layer after each fully connected layer decreases the performance for the VGG16 network, especially for the RMSProp optimizer. For the VGG19 network, the performance did not change significantly when considering the addition of dropout layers. The performance of the InceptionV3 network was affected by the inclusion of the dropout layer, for both the Adam and RMSprop optimizers. The ResNet network was the only network

that benefited from adding the dropout layer, and the performance of RMSProp with dropout was higher with respect to the Adam optimizer. The VGG architectures were the best performers among the different competitors taken into account, and they were outperformed only by our proposed architecture. InceptionV3 and ResNet were the poorest performers. For the InceptionV3 model, we can speculate that the inception module did not increase the performance, while adding too many layers is not beneficial for the ResNet model. All in all, our proposed architecture achieved higher results for both Adam and RMSprop optimizers.

5.6. Comparison between Different State-of-the-Art CNN

Arjmand et al. [22] presented a network with three convolution layers, where the number of kernels is 64, 32, and 16, respectively. The kernels size used was 5×5 , 3×3 , and 3×3 , respectively. Every convolution layer was followed by a batch normalization layer and max-pooling layer, except for the third convolution layer, which was followed by a batch normalization layer only. There are two dropout layers, one after the second convolution layer and the second before the fully connected layer. Compared to our network, Arjmand architecture was the second-best architecture, among the set of competitors, for both RMSprop and Adam optimizers. The authors placed the normalization layer before the activation layer and the dropout layer in the middle of the convolution blocks. The main difference between our proposed architecture and the architecture proposed by Arjmand et al. [22] relies on the number of convolution layers used. From the results obtained, it seems that the choice of the number of convolution layers plays a fundamental role in the performance of the model. In particular, a network with just three convolution layers seems to be not sufficient for dealing with the complexity of the application at hand. The architecture of Arjmand et al. [22] is shown in Figure 7.

Lai et al. [24] introduced a network with six convolution layers and no fully connected layers. This network achieved the best results, among the competitors of our proposed architecture, for both the optimizers. The kernel size used in both the convolution layers and the pooling layers is the highest compared to all the tested CNNs. No dropout layers or batch normalization layers were used. The results of Adam and RMSProp optimizers are comparable. The differences between this network and our network are the size of the kernels used (which is higher than our proposed network), the absence of any regularization layers, the absence of any fully connected layers, and the number of convolutional layers, (which is higher in our model). The main difference in terms of design choices between the architecture of Lai et al. [24] and our architecture is the absence of any regularization layers from their architecture. According to the experiments we performed, this seems to be the main cause for the lower performance of the network proposed by Lai and coauthors. Figure 4 shows the architecture of Lai et al. [24].

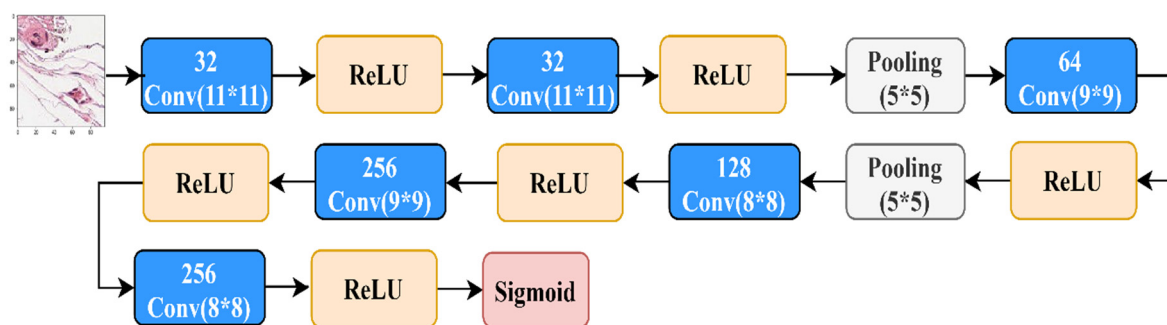


Figure 4. The architecture of Lai et al. [24].

Nguyen et al. [20] presented a network with five convolution layers, one fully connected layer, three dropout layers, and six normalization layers. The authors placed the normalization layer after the activation layer, and the dropout layer in the middle of the convolution blocks. This is the only network with an activation function other than the ReLU function. The results of Adam and RMSProp

optimizers are different, with Adam being the best optimizer. Compared to our proposed network, the main differences are the number of convolution layers, the presence of the normalization layer after the activation function, and the usage of only one fully connected layer. The difference between the performance of our proposed architecture and the architecture presented by Nguyen et al. [20] can be explained by the number of convolution layers (that is lower with respect to our architecture) and by the position of the normalization layer after the activation layer. In particular, placing the normalization layer after the activation function was not suggested in the original paper that introduced the normalization layer [38]. Figure 5 shows the architecture of Nguyen et al. [20].

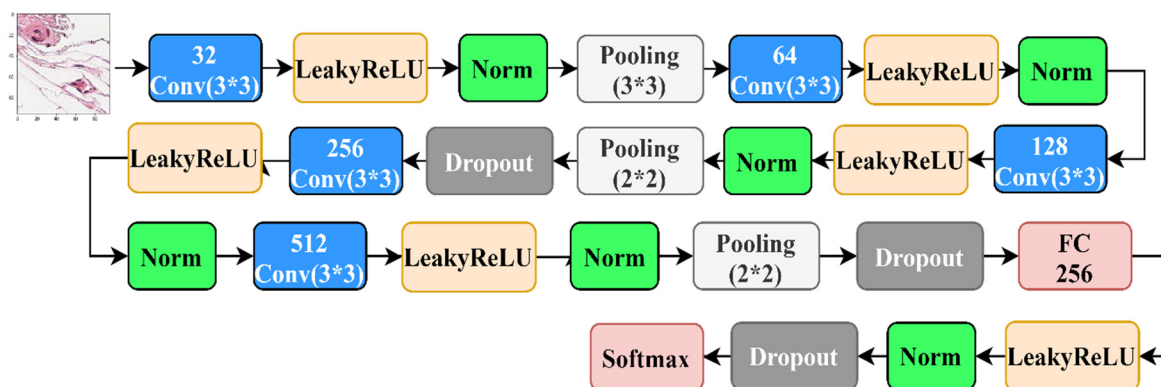


Figure 5. The architecture of Nguyen et al. [20].

Basha et al. [25] designed an architecture with 4 convolution layers and two fully connected layers. The authors also placed the normalization layer after the activation layer. Moreover, the authors placed a normalization layer in the classifier block. The results of Adam and RMSProp optimizers are significantly different, with RMSProp being the best optimizer. Compared to our architecture, the main differences are the number of convolution layers used (we used 15 instead of 4), the presence of the normalization layer after the activation function, and the usage of the normalization layer in the classifier block. The difference between the performance of our proposed architecture and the architecture proposed by Basha et al. [25] can be due to the number of the convolution layers, placing the normalization layer after the activation layer, and the usage of dropout layers in the classifier block. Figure 6 shows the architecture of Basha et al. [25].

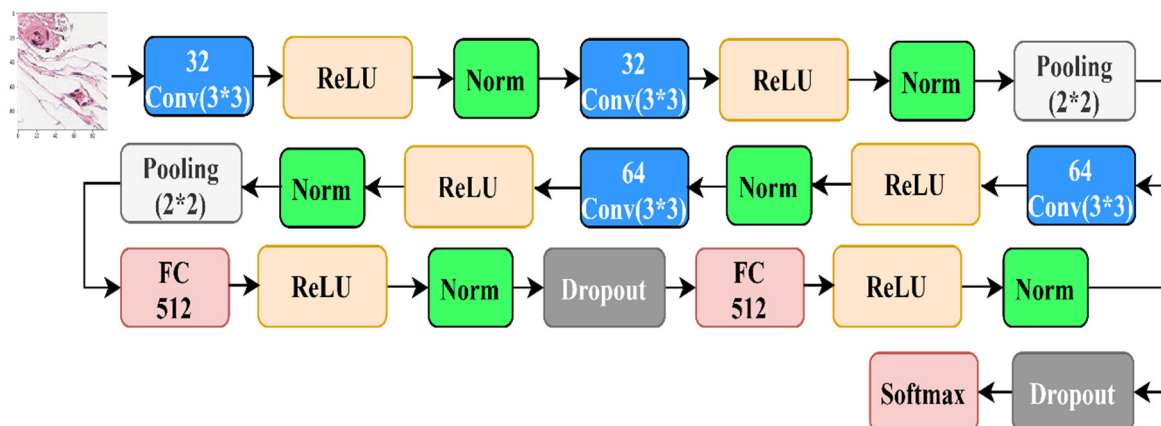


Figure 6. The architecture of Basha et al. [25].

The architecture of Bayramoglu et al. [21] has three convolution layers and two fully connected layers. The authors placed the normalization layer after the activation layer and after the pooling layer as well. The results of Adam and RMSProp optimizers are significantly different, with Adam being

the best optimizer. Compared to our design, this architecture has low capacity, since it has only three convolution layers. The difference between the performance of our proposed architecture and the architecture proposed by Bayramoglu et al. [21] can be explained (beyond the number of convolution layers) considering the usage of a larger kernel size in the first convolution layer. This is something that could be detrimental to the performance of the network. Using smaller kernel sizes allows the network to learn complex, more non-linear features. The architecture of Bayramoglu’s [21] is shown in Figure 8.

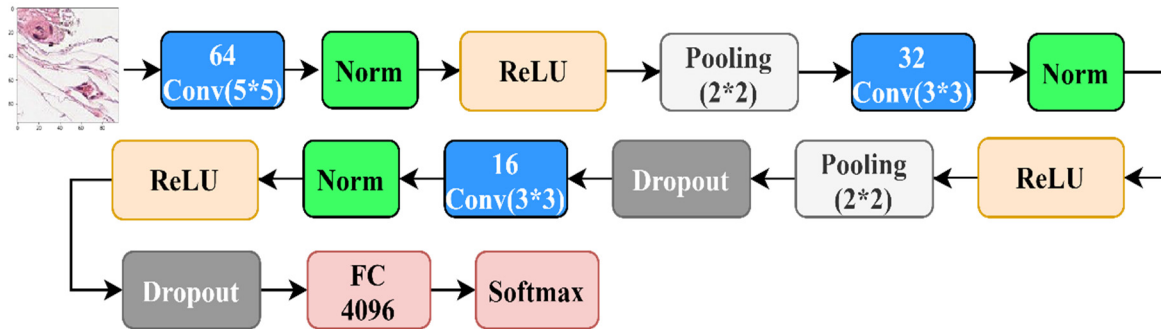


Figure 7. The architecture of Arjmand et al. [22].

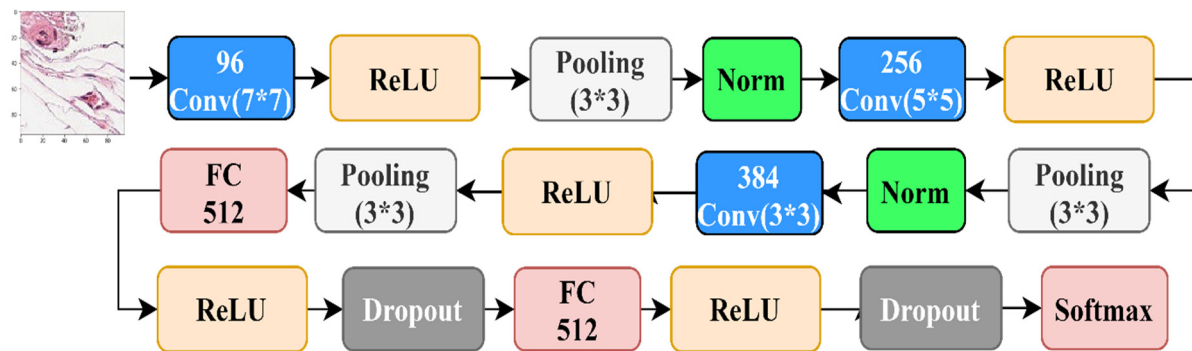


Figure 8. The architecture of Bayramoglu et al. [21].

Sirinukunwattana et al. [23] proposed a network with only two convolution layers and two fully connected layers. The network is the worst performer in our comparison, probably because it has a very low capacity compared to the other networks taken into account. The results of Adam and RMSProp optimizers are significantly different, with RMSProp being the best optimizer. In this network, the ReLU activation function were used, and two pooling layers were placed after each convolution layer. Figure 9 shows the architecture of Sirinukunwattana’s [23].

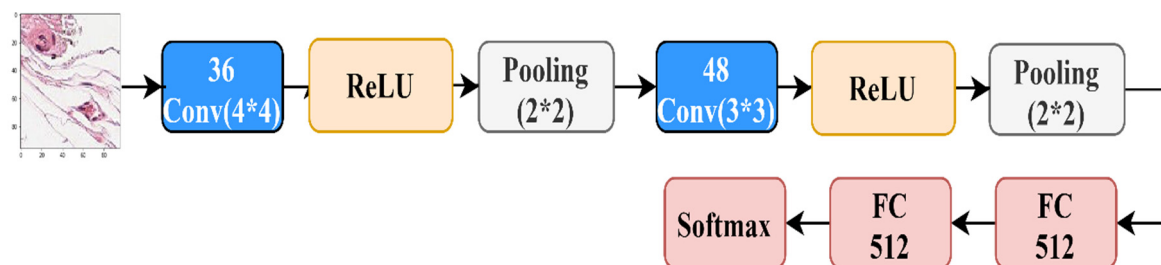


Figure 9. The architecture of Sirinukunwattana et al. [23].

6. Conclusions

In this paper, we introduced a novel CNN architecture that is designed to classify histopathology images. The training and evaluation of the architecture were conducted on the publicly available

PatchCamelyon dataset. The proposed architecture has fifteen convolution layers and two fully connected layers. The highest AUC obtained using our architecture was 95.46%. In this work, we have also studied the effect of different activation functions on the CNN performance and the effect of the location of the activation function on the performance of the network. Based on the obtained results, two main points must be highlighted: We recommend authors to try different activation functions and to fully analyze their impact other than choosing the ReLU activation function as a default. Based on our results, we recommend placing the normalization layer before the activation function. We do encourage researchers to examine our proposed CNN on different datasets and report the performance achieved.

We acknowledge several limitations in this work that can be addressed in future work. First, the performance of the proposed model slightly increases with respect to the models that were presented in the literature. Second, the proposed model was inspired by the VGG16 architecture, and the differences are the positioning of the regularization layers, the activation functions used, and the number of neurons in the fully connected layers. We believe that the two aforementioned limitations could be overcome through the use of neuroevolution algorithms that can provide a different way of exploring the search space of deep learning architectures. Finally, the human performance was reported on the Camelyon dataset, which is a reduced dataset with respect to the PatchCamelyon dataset used in this study. Thus, it would be interesting to assess the human-experts' performance on this bigger dataset.

Author Contributions: "I.K. and M.C. conceived and designed the experiments; I.K. performed the experiments; I.K. analyzed the data; M.C. contributed to the analysis; I.K. and M.C. wrote the paper. M.C. supervised the work and provided funding". All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by FCT (Fundação para a Ciência e a Tecnologia), Portugal and the Slovenian Research Agency, Slovenia. The APC was funded by FCT.

Acknowledgments: This work was supported by FCT (Fundação para a Ciência e a Tecnologia), Portugal, through funding of the projects GADgET (DSAIPA/DS/0022/2018) and AICE (DSAIPA/DS/0113/2019). Mauro Castelli also acknowledges financial support from the Slovenian Research Agency (research core funding No. P5-0410) and the support from NVIDIA Corporation.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Siegel, R.L.; Miller, K.D.; Jemal, A. Cancer statistics. *CA. Cancer J. Clin.* **2020**, *70*, 7–30. [[CrossRef](#)] [[PubMed](#)]
2. World Health Organization. Cancer 2018. Available online: <https://www.who.int/news-room/fact-sheets/detail/cancer> (accessed on 12 February 2020).
3. He, L.; Long, L.; Antani, S.; Thoma, G. Histology image analysis for carcinoma detection and grading. *Comput. Methods Programs Biomed.* **2012**, *107*, 538–556. [[CrossRef](#)] [[PubMed](#)]
4. Robbins, P.; Pinder, S.; de Klerk, N.; Dawkins, H.; Harvey, J.; Sterrett, G.; Ellis, I.; Elston, C. Histological grading of breast carcinomas: A study of interobserver agreement. *Hum. Pathol.* **1995**, *26*, 873–879. [[CrossRef](#)]
5. Metter, D.; Colgan, T.; Leung, S.; Timmons, C. Trends in the US and Canadian Pathologist Workforces From 2007 to 2017. *JAMA Netw. Open* **2019**, *2*, e194337. [[CrossRef](#)]
6. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.* **1989**, *1*, 541–551. [[CrossRef](#)]
7. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [[CrossRef](#)]
8. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [[CrossRef](#)]
9. Luo, H.; Yang, Y.; Tong, B.; Wu, F.; Fan, B. Traffic Sign Recognition Using a Multi-Task Convolutional Neural Network. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 1100–1111. [[CrossRef](#)]

10. Sermanet, P.; LeCun, Y. Traffic sign recognition with multi-scale Convolutional Networks. In Proceedings of the 2011 International Joint Conference on Neural Networks, San Jose, CA, USA, 31 July–5 August 2011; pp. 2809–2813.
11. Kim, Y. Convolutional Neural Networks for Sentence Classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1746–1751.
12. Zhang, X.; Zhao, J.; LeCun, Y. Character-Level Convolutional Networks for Text Classification. In Proceedings of the 28th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2015; Volume 1, pp. 649–657.
13. Abdel-Hamid, O.; Mohamed, A.; Jiang, H.; Deng, L.; Penn, G.; Yu, D. Convolutional Neural Networks for Speech Recognition. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2014**, *22*, 1533–1545. [[CrossRef](#)]
14. Abdel-Hamid, O.; Mohamed, A.; Jiang, H.; Penn, G. Applying Convolutional Neural Networks concepts to hybrid NN-HMM model for speech recognition. In Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 25–30 March 2012; pp. 4277–4280.
15. Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; Dauphin, Y. Convolutional Sequence to Sequence Learning. In Proceedings of the 34th International Conference on Machine Learning (ICML 2017), Sydney, Australia, 6–11 August 2017.
16. Gehring, J.; Auli, M.; Grangier, D.; Dauphin, Y. A Convolutional Encoder Model for Neural Machine Translation. *arXiv Prepr.* **2017**, arXiv:1611.02344.
17. Veeling, B.S.; Linmans, J.; Winkens, J.; Cohen, T.; Welling, M. *Rotation Equivariant CNNs for Digital Pathology BT—Medical Image Computing and Computer Assisted Intervention—MICCAI 2018*; Springer Cham: Granada, Spain, 2018; pp. 210–218.
18. Ehteshami Bejnordi, B.; Veta, M.; Johannes van Diest, P.; van Ginneken, B.; Karssemeijer, N.; Litjens, G.; van der Laak, J.A.W.M.; the CAMELYON16 Consortium, Hermsen, M. Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Cancer. *JAMA* **2017**, *318*, 2199–2210. [[CrossRef](#)] [[PubMed](#)]
19. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [[CrossRef](#)]
20. Nguyen, P.T.; Nguyen, T.T.; Nguyen, N.C.; Le, T.T. Multiclass Breast Cancer Classification Using Convolutional Neural Network. In Proceedings of the 2019 International Symposium on Electrical and Electronics Engineering (ISEE), Ho Chi Minh, Vietnam, 10–12 October 2019; pp. 130–134.
21. Bayramoglu, N.; Kannala, J.; Heikkila, J. Deep learning for magnification independent breast cancer histopathology image classification. In Proceedings of the 2016 23rd International Conference on Pattern Recognition (ICPR), Cancun, Mexico, 4–8 December 2016.
22. Arjmand, A.; Angelis, C.T.; Tzallas, A.T.; Tsipouras, M.G.; Glavas, E.; Forlano, R.; Manousou, P.; Giannakeas, N. Deep Learning in Liver Biopsies using Convolutional Neural Networks. In Proceedings of the 2019 42nd International Conference on Telecommunications and Signal Processing (TSP), Budapest, Hungary, 1–3 July 2019; pp. 496–499.
23. Sirinukunwattana, K.; Raza, S.E.A.; Tsang, Y.; Snead, D.R.J.; Cree, I.A.; Rajpoot, N.M. Locality Sensitive Deep Learning for Detection and Classification of Nuclei in Routine Colon Cancer Histology Images. *IEEE Trans. Med. Imaging* **2016**, *35*, 1196–1206. [[CrossRef](#)] [[PubMed](#)]
24. Lai, Z.; Deng, H. Medical Image Classification Based on Deep Features Extracted by Deep Model and Statistic Feature Fusion with Multilayer Perceptron. *Comput. Intell. Neurosci.* **2018**, *2018*, 1–13. [[CrossRef](#)] [[PubMed](#)]
25. Basha, S.H.S.; Ghosh, S.; Babu, K.; Dubey, S.; Pulabaigari, V.; Mukherjee, S. RCCNet: An Efficient Convolutional Neural Network for Histological Routine Colon Cancer Nuclei Classification. In Proceedings of the 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, 18–21 November 2018.
26. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
27. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z.B. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 2818–2826.

28. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
29. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10), Haifa, Israel, 21–24 June 2010.
30. Maas, A.L.; Hannun, A.Y.; Ng, A.Y. Rectifier nonlinearities improve neural network acoustic models. In Proceedings of ICML 2013, Atlanta, GA, USA, 16–21 June 2013.
31. Clevert, D.-A.; Unterthiner, T.; Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv Prepr.* **2016**, arXiv:1511.07289.
32. Klambauer, G.; Unterthiner, T.; Mayr, A.; Hochreiter, S. Self-Normalizing Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; pp. 971–980.
33. Kingma, D.; Ba, J. Adam: A Method for Stochastic Optimization. *Int. Conf. Learn. Represent.* **2014**, arXiv:1412.6980.
34. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv Prepr.* **2016**, arXiv:1609.04747.
35. Shorten, C.; Khoshgoftaar, T.M. A survey on image data augmentation for deep learning. *J. Big Data* **2019**, *6*, 60. [[CrossRef](#)]
36. Chollet, F. *Keras*; GitHub: San Francisco, CA, USA, 2015.
37. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrada, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv* **2015**, arXiv:1603.04467.
38. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv Prepr.* **2015**, arXiv:1502.03167.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).