Project - CM2003

Classification of images from histopathologic scans of lymph node sections: A comparison of different DNN architectures

Joakim Stoor

October 2019

1 Introduction

Deep learning is still in its infancy, and the true power of this technology is still to be discovered. Yet, tremendous strides have been taken since 2012 when AlexNet became the first deep neural network (DNN) to defeat conventional computer vision methods in a large-scale image classification challenge [1].

Some key factors behind the recent success of deep learning are the following: increased accessibility to more and better data, increased computational efficiency via the usage of GPU computing, and deeper networks [1]. Also, the success has been facilitated by a rich online community and by the evolution of deep learning frameworks that offers building blocks for design, training, and validation.

In this paper, several DNN architectures are trained and validated on the PatchCamelyon datatset [2]. The goal was to compare the performances of each network on the dataset, and to asses if better performance could be reached by producing an ensemble learner from the individual networks.

The PatchCamelyon dataset contains images from histopathologic scans of lymph node sections. It consists of 327.680 color images of size 96×96. It's a binary classification task where a positive label indicates that the center 32×32 region of the patch contains metastatic tissue [2]. The dataset is divided into a training, validation, and test set consisting of 2^{18}, 2^{15} and 2^{15} images. All sets are class balanced.

2 Architectural developments

Mirroring the progress in other scientific fields, the evolution of convolutional neural netoworks (CNNs) has been one where researchers have redefined the basic building blocks of networks by increased encapsulation and abstraction. In AlexNet, the basic building block is a convolutional layer followed by a non-linear activation function and max-pooling [1]. In VGG, a block consists of a sequence of such convolutional layers and activations, followed by max-pooling [1]. Both of these networks uses a convolutional part for feature selection followed by an MLP for classification.

As networks become deeper, the problem of vanishing gradients increases. This can be addressed by introducing residual blocks with skip connections, and it's the main idea behind residual networks (ResNets). A residual block is like a normal convolutional block with the addition of a shortcut connection between the input and the last activation function of the block [1]. The two branches of the block are added before the last activation function is applied. In these networks, the highest abstraction level is the residual module, which consists of residual blocks. Global average pooling is applied before a dense layer at the end of the network to avoid using several fully connected layers.

A natural extension of ResNets is to allow for even more dense residual connections. In densely connected networks (DenseNets), the main differences from ResNets is the use of skip connections between each layer within a block, and that branches are concatenated instead of added [1]. The growth of the number of feature channels for a network input is thus defined by a growth factor that defines the number of convolutional filters added after each layer. Finally, DenseNets uses transition layers between dense blocks to reduce the complexity of the model.

Further architectural developments and refinements beyond those listed here have been made over the years. Nonetheless, for this paper, the mentioned improvements will suffice.

3 Solution strategies

The code for this project was written in python using TensorFlow and Keras. It can be found on my GitHub page [3] where I also describe the practical details of the project, including the file-system.

3.1 Network architectures

VGG16, ResNet50 and DenseNet121 were trained and validated on the PatchCamelyon dataset. The standard off-the-shelf convolutional filter configuration of these networks were initially optimized for ImageNet images [1]. Thus, for this project, the filter configurations in the different layers were changed and customized to accommodate the images in the PatchCamelyon dataset.

For the ResNet and DenseNet implementations, the first convolutional layer ordinarily consists of filters of size 7×7 with stride 2 [1], but to factor in the smaller PatchCamelyon images, these were changed to 5×5 with stride 3. Also, the number of convolutional filters in the first layer of each network, including VGG, was changed from 64 to 16. See [3] for the specific network settings.

3.2 Training and validation

Each network was individually trained on the training set and its hyperparameter settings were chosen based on the network's performance on initial tests on the validation set. The best model was saved during training by monitoring and measuring the validation loss. Binary cross entropy was used as the loss function, and binary accuracy, precision and sensitivity were logged and used as evaluation metrics. An additional ensemble learner, using the mean predictions from the three networks, was created after training.

Data augmentation was applied to the training set for each network. Images were randomly flipped horizontally and vertically, and translated with a distance up to four pixels in the x- and y-direction. All images were normalized to have values between zero and one.

The different "networks" were then used for evaluation on the test set. They were compared on all the metrics used during training, but also on the reciever operating characteristic (ROC) curve and area under curve (AUC) score that were calculated for each learner.

4 Findings

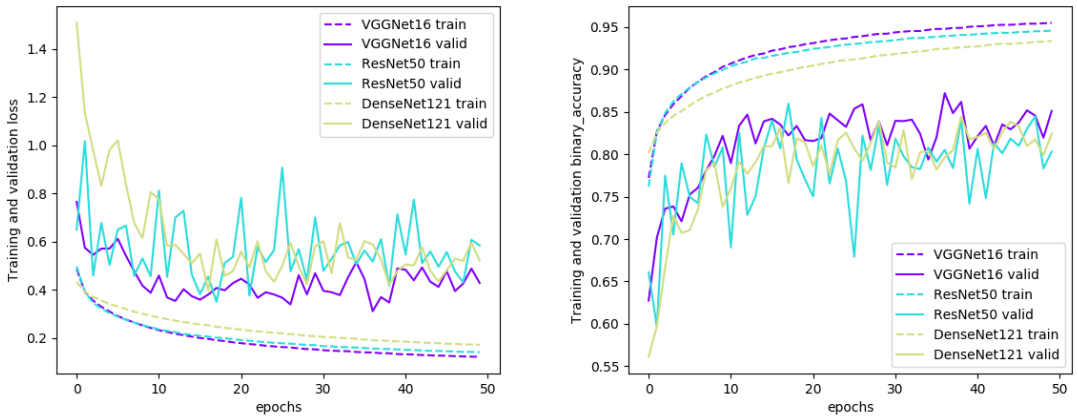
The results in this section applies to specific hyperparameter settings found through a non-exhaustive search in hyperaparameter space for each network.

4.1 Training

By performing initial tests, it was found that smoother training and better validation scores were achieved by reducing the number of feature channels in the first convolutional layer of each network. The base value was thus reduced from 64 to 16. Generally, it was difficult to achieve a regularizing effect that minimized the discrepancy between the training and validation loss and that simultaneously produced good validation scores (see figure 1). A more thorough hyperparameter search would be needed to analyze this further.

It can be observed that the ResNet and DenseNet variants, i.e. the more expressive networks, performed somewhat worse than the VGG variant on the validation set. DenseNets and ResNets

uses global average pooling operations, but only the center regions of the images in the PatchCamelyon dataset are used during labeling, so this operation may be sub-optimal for the classification task at hand. Averaging over the spatial dimensions of a tensor might be detrimental as essential "locality" information is lost.



(a) binary cross entropy loss

(b) accuracy metric

Figure 1: Learning curves for the training and validation sets. The loss function in use was binary cross entropy. The training curves are dashed for each network and the validation curves are solid lines.

4.2 Evaluation

The networks that achieved the lowest validation loss scores during training were saved and applied to the test and validation sets. The results can be seen in table 1. Aside from the performance metrics used during training, ROC curves (figure 2) and AUC scores were calculated for each learner on each set.

	accuracy	AUC	NLL	precision	sensitivity
VGG	81.5 (87.2)	89.2 (94.8)	.562 (.312)	61.9 (62.4)	45.7 (55.1)
ResNet	76.8 (86.0)	85.7 (93.4)	.706 (.350)	60.9 (61.8)	43.8 (56.9)
DenseNet	74.8 (83.1)	87.9 (93.4)	.680 (.397)	61.7 (62.7)	39.1 (50.6)
Ensemble	78.8 (88.0)	90.4 (95.4)	.495 (.289)	84.5 (85.6)	61.7 (76.8)

Table 1: Loss and metric scores for the networks on the PatchCamelyon test and validation sets. The test scores are shown regularly and the validation scores are within parentheses. AUC is the area under curve measure and NLL is the negative log loss, i.e. the binary cross entropy loss.

The validation set scores didn't generalize that well to the test set, pointing to a discrepancy between the underlying probability distributions for the data within the different sets. Preferably,

the validation data could have been added to the training data, to use all the data with k-fold cross validation to get a more stable estimate of the prediction error.

As of this moments, the benchmark performance on the dataset was set by a rotation equivariant DenseNet [2]. It achieved an accuracy score of 89.8, a binary cross entropy score of 0.260 and an AUC score of 96.3. The scores from the implemented networks in table 1 are pretty far off from these scores, especially those produced by the networks that uses global average pooling.

Partially due to the discrepancy in performance between the three implemented networks, the ensemble learner didn't achieve a better accuracy score on the test set than the best-performing VGG16 variant, but it did achieve higher AUC, precision and sensitivity scores. Notably, on the validation set, where the performance scores differed less between the networks, the ensemble learner performed better on all metrics.

While keeping the aforementioned results in mind, the ROC curves and the AUC scores tells us that even though benchmark performance wasn't achieved, the binary classifiers are much more capable of distinguishing between the two classes than a random classifier.

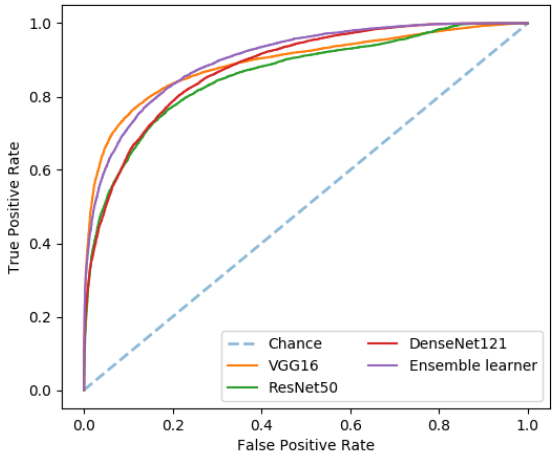


Figure 2: Reciever operating characteristic (ROC) curves for the classifiers on the test dataset.

5 Conclusions

Further hyperparameter testing is needed to optimize the networks on the PatchCamelyon dataset. The effects of average pooling needs further investigation to reach conclusions about how severe the consequences of this operation are on the final results. As of now, the VGG16 variant reached best performace on the dataset, and no improvement was achieved by implementing the ensemble learner.

References

[1] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. 2019. http://www.d2l.ai.

[2] Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant CNNs for digital pathology. June 2018.

[3] Joakim Stoor. Classification of images extracted from histopathologic scans of lymph node sections. https://github.com/jstoor-elf/CM2003-9002/tree/master/Project/, 2019 (accessed October 16, 2019).