Convolutional Neural Network Performance on Pneumonia Identification

Henry H. Lee
PID: A14741955
hhl045@ucsd.edu

June 15, 2019

Abstract

This paper presents the performance of three models on classifying pneumonia x-rays. These models are the pretrained ResNet-101, VGG19, and DenseNet161. These models are used by fine tuning them to the x-ray images. A central point of this dataset is that it has a low amount of examples (5000) that are unbalanced. It is the performance of these models in a binary classification task under suboptimal conditions that is of special interest.

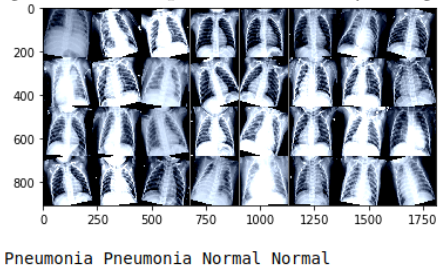
1 Introduction

Convolutional neural networks are neural network architectures designed for images[1] that has demonstrated constant improvement over the years and have reached a point that is sufficient to be applied on many real world problems. Some incentives for leveraging this technology includes time, money, and precision. Humans with the expertise to understand a complex problem relying on visual cues often take a considerable amount of time to determine what the answer to the problem is. However a machine program can suggest a solution almost instantly. Following this point, money is saved by aiding the expert through the immediate suggestion of a possibility to consider. Lastly, there is the potential for the machine classifier can be potentially better than humans in precision. Many visual problem with sufficient data can leverage convolutional neural networks today to produce impressive results.

The problem this paper addresses with these CNNs is a collection of chest x-rays of pneumonia patients found on Kaggle [2]. Specifically, the aim is to be able to diagnose the condition based on an x-ray image with minimal false positives and false negatives. The training set contains 3875 examples of pneumonia and 1341 examples of a chest x-ray without the condition. In order to address this imbalance, the images were duplicated. This was done by creating duplicate files due to time constraints so the oversampling isn't exactly ideal. However, it

produced sufficient results. The validation set is a small amount- only 19, which was discovered late into the project. Finally, the test set amounts to 624.

Figure 1: Examples of chest xray images



2 Methods

This section details the approach towards fine tuning the models towards this specific problem was implemented. The general idea for a single model is to load the pretrained model, find the dimensions of the final layer, and reinitialize the layer for the case of 2 classes. This prepared model is then trained and tested using the same function with the optimizer being the only thing that is changed after the final layer reinitialization.

The aim of using pretrained models is to fine tune them to this specific problem. This can be done by inspecting the parameters of the model and changing the final layer such that it results in an output of two nodes. In other words, this new final layer can leverage the already capable parts of the network to solve this classification problem [3]. All computation was performed using a Nvidia GTX-1080ti.

2.1 Training

Training was done with 21 epochs with batch sizes of 32, which were an arbitrary numbers. The epochs in particular proved to be excessive. Most models trained close to the final model by the 13th iteration. This is likely due to the relatively small size of the dataset at 7715 images after oversampling, as there is only so much that the models can learn from. The lowest validation score was selected to be the metric for which iteration would be used as the final model, as it is desired that the model be able to generalize well to images it has yet to see.

To improve the robustness of the fine tuned models, the image augmentation is applied so that there is more variability in their orientation and brightness [4]. After the images are augmented, the resulting tensor is normalized to the values mean = [0.485, 0.456, 0.406] and std = std = [0.229, 0.224, 0.225] to keep inputs consistent with those used to pretrain it with ImageNet. If these inputs are not normalized, it is likely that the model will no longer be able to properly

detect edges and shapes as it should, as the values will be far too different from what it is trained to detect. This was discovered in the next section of testing.

2.2 Testing

Because the dataset is imbalanced, precision and recall is used to evaluate the models. The models make predictions using the test set and are compared to the labels to create a confusion matrix. Precision is of particular interest here, as a diagnosis desired to be correct. In the case of a false positive, it causes the patient unnecessary mental strain. In the case of a false negative, a patient will go untreated.

A failure to use pretrained models properly occured in this project which revealed an aspect of how models work. In the case of a model trained with a specific normalization, he failure to normalize the input will trivialize its ability to classify. Initially, the transformer for the test data lacked a normalization on the result tensors. In this case of this project, it was truly trivial, as the classifiers would opt to determine that every single image is a case of pneumonia. Consistent normalization of all inputs into a model should be done so that there isn't the temptation to discard the project due to the failure of the model on inputs it did not expect.

3 Experimentation

Experimentation was done in two ways centered around the nature of the dataset to evaluate the models. Naturally, since the dataset is a collection of images in the form of jpg files, convolutional neural network architectures were chosen. Since the dataset was relatively small, this project also aimed to compare the raw performance of two optimizers. This was all implemented using Pytorch and its computer vision package torchvision.

3.1 Architectures

Three models that had the some of the best top one accuracy on ImageNet was selected for use in this project. These models are the pretrained ResNet101 [5], DenseNet161 [6], and VGG19 [7] models were loaded from the torchvision package. By loading the pretrained models, the weights will be initialized closer to values that yare relevant to generating useful information from the start.

On the part of the three architectures, in how well they would perform if the last layer is adapted to the task. This layer is changed to two opposite outputs, one for a normal x-ray and one for a pneumonia x-ray.

3.2 Stochastic Gradient Descent vs Adam Optimizer

To see the impact of different optimzers. These optimizers are Stochastic Gradient Descent with a learning rate of .0001 and momentum of .9 and Adam with

a learning rate of .0001 and a weight decay of .00001. These optimizers were chosen at random. The learning rate was originally .001 for stochastic gradient descent, but was changed and retrained to match the learning rate for the Adam optimizer training.

4 Results

Here we have the results of training and running these models using the dataset. The models all achieve a high recall by avoiding false negatives. Precision, however, is the most useful metric in this scenario as explained above. In this metric, DenseNet161 and VGG19 perform the best with a precision of .89.

In terms of the optimizers, Adam increases the training time for a slight decrease in loss. In this scenario, this is actually very desirable since every decision made by the machine matters. Training time is a cheap price to pay if this model were to be actually used to diagnose a large number of patients.

Model	Optimizer	Time	Best Val	Recall	Precision
ResNet101	SGD	**36m 11s**	0.072004	.99	.85
DenseNet161	SGD	44m 37s	0.037080	**1.00**	. 85
VGG19	SGD	40m 15s	0.043354	.99	.83
ResNet101	Adam	45m 21s	0.004875	**1.00**	.84
DenseNet161	Adam	49m 11s	0.019972	.98	**.89**
VGG19	Adam	45m 39s	**0.003197**	.98	**.89**

Table 1: Model Results

5 Discussion

The general trend from the results is that VGG19 performs the best on classifying chest x-rays for pneumonia. This is a surprising result, as DenseNet and ResNet perform better on ImageNet, so it was expected that they would perform better on this specific task also. However, it turns out that better performance on one general task does not necessarily mean that it will perform better on all tasks.

Originally, feature extraction, was attempted on this problem by only updating the parameters of the final layer. These trial trained in half as much time, however the loss failed to decrease at a satisfactory pace if at all. The implementation was likely fault. For this reason, the feature extraction models were discarded. However, the two extremes of one layer being updated and all layers updating points to the potential for selectively updating certain layers for the advantages of both.

In terms of general results, these convolution neural network models prove themselves to perform quite well despite the limitations of the dataset. It was

expected that between the imbalanced training data, small validation set, and relatively small size of the dataset overall that the models would struggle to reach even .8 precision. This reveals that at least of binary image classification tasks, collecting immense amounts of data is not always necessary.

6 Conclusion

The unexpected result highlights the fact that it is important to know how to experiment with models and architectures to cater to a specific task. Simply choosing the architecture with the state of the art results and using it without question for any task fails to optimally solve problems in the real world. when facing problems that are sensitive to error, such as medical diagnoses, it is crucial to be as precise as possible. Applying deep learning to solve these kinds of problems has potential to improve the world for humans, but we should take care to not accidentally become machines ourselves.

Acknowledgments

Much appreciation to Paul Mooney on Kaggle for providing the data in an accessible way and Nathan Inkawhich [8] for creating a great tutorial on fine tuning torchvision models.

References

[1] LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. *Nature*, 521(7553), pp.436-444.

[2] Gulshan, V., Peng, L., Coram, M., Stumpe, M.C., Wu, D., Narayanaswamy, A., Venugopalan, S., Widner, K., Madams, T., Cuadros, J., et al. Development and alidation of a deeplearning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*, 2016.

[3] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *In Advances in Neural Information Processing Systems*, 2014.

[4] Wang, Jason and Perez, Luis. The effectiveness of dataaugmentation in image classification using deep learning. *Stanford University research report*, 2017.

[5] He, Kaiming et al. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp.770-778.

[6] Huang, G., Liu, Z., and Weinberger, K.Q. (2017). Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2261-2269.

[7] Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556.

[8] Nathan Inkawhich: Finetuning Torchvision Models, `https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html`