McGill University

Mini Project 3: Multi-label Classification of Image Data

COMP 551: Applied Machine Learning

Hassan Gaber (260891600)
Anthony Porporino (260863300)
Felix Simard (260865674)

Group 19

Nov 25th, 2021

Abstract

In this third mini project, our task was to develop a model to classify image data. More specifically, our model should distinguish which combination of handwritten digit and alphabetical letter was drawn in each input image. To evaluate the performance of our model, we took part in a Kaggle competition where we competed for the highest test accuracy. Our solution obtained 91% on the public portion of the test set. We found that the architecture of the neural network and bootstrap aggregation were the two main factors that determined the accuracy of our model. Using a variation of a convolutional neural networks called AlexNet [1] and bootstrap aggregation, we were able to balance our model's high variance with a decent bias.

Introduction

The main task of this project was to design a neural network that could achieve the highest possible score in the Kaggle competition. We were given 60,000 images as training data, half of them containing correct labels and were tested on 15,000 images from a test set. Our team investigated 3 different CNNs. The first being a simple model based on the PyTorch tutorial for this course. The second and third models are famous CNNs that achieved state of the art results (for their time) on the classic ImageNet problem. They are AlexNet and VGG11. We found that AlexNet performed the best on this problem. The simple model did not achieve good results on either training or validation set and VGG11 was a very deep model that also did not perform well. AlexNet, with some adjustments, proved to be a model that had a low bias score and high variance. We successfully used bootstrap aggregation to decrease the variance of the model. We also attempted to label the unlabelled dataset by choosing entries where all 5 of our "bootstrap models" agreed. This new set was then used as more training data. Lastly, we preprocessed the data before training. Neither of these methods, however, increased the performance of our model.

Datasets

The provided datasets contained images that showed artificially created pictures of handwritten numbers and letters. The letter and number contained in each image could be positioned anywhere in the picture. We were given 60,000 training instances, of which only 30,000 had labels associated to them. For testing, our task was to apply our model to 15,000 unlabelled test instances and report our predictions to Kaggle. The data was preprocessed in two different ways, firstly, by filtering the image and making important features clearer. Secondly, by distorting the image or augmenting it such that important features are harder to identify. Unintuitively, the image filtering decreased validation and test set accuracy indicating that the CNN had activated incorrect weights. This can be attributed to the fact that the image data is inherently distorted and training on clearer images would not help the model learn how to classify low-quality images. The augmented data obtained using Torchvision transforms increased training time and had lower scoring accuracies than models trained with non-augmented data. Our implementation of bagging created 5 different datasets from the 30,000 labelled images. The new dataset entries were chosen randomly with replacement from the original dataset. This means that many of the items in each dataset are duplicates. This is intended since this allows the datasets to be unique. Once the datasets were extracted, they were each used to train a model using our AlexNet inspired CNN. By using these 5 trained models to label the unlabelled dataset, we can aggregate the 5 predictions to create labels for each entry. We kept only the labels where all 5 models agreed, and this resulted in a new labelled set of 16,684 images. We manually looked through 40 images and labels and all 40 were classified correctly.

Results

As mentioned, we trained using 3 different models. The first simple model gave an accuracy of less than 1% on validation and training data. This showed us that the bias of the model is too high, and we needed a more complex model. Using the VGG11 implementation, our model was now very deep containing 8 convolutional layers. This model's performance was around 40-50% and indicated that the bias was still very high. We finally designed a model that performed very well on the training set having accuracies around 90%-95% on the training datasets. This model was a variation of AlexNet. It contains 5 convolutional layers each with a RELU activation function and it also uses MaxPooling. This section is followed by 2 very wide linear layers and dropout layers. For this model, the Adam optimizer yielded the best results using a learning rate of 1×10^{-3} with a batch size of 64 trained on 10 epochs. Smaller batch sizes took many more epochs to converge, usually at around 30 epochs. *Figure* 1, from the Appendix, shows a graph depicting the validation accuracy versus the batch size. Despite taking longer to converge, as shown in *Figure 1*, a smaller batch size did not yield good validation accuracies. Larger batch sizes also performed worse on the validation set. We chose a batch size of 64 as this proved to offer the highest accuracy on the validation set. *Figure 2* shows the training loss and validation loss vs number of epochs when training our model on 70% of the labelled dataset.

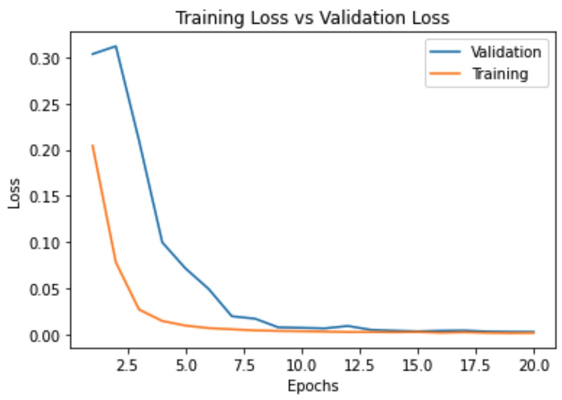

Figure 1: Train and Validation Loss vs Epoch size

The best performing model finished training at loss values of around 2×10^{-3} indicating the gradient had converged to the global maximum of the objective function. The validation accuracy however was considerably less than the training accuracy and was around 84-86%. We attempted to train on preprocessed images however this resulted in a decrease in both training and validation accuracies. We also attempted to train on more data using the 16,684 new images derived from the unlabelled dataset. Unfortunately, this did not increase accuracy either, rather it stayed the same as it was previously. The key result we found was that bootstrap aggregation significantly decreased the variance of our model. It allowed us to increase our test and validation accuracy from 86% to 91%. As mentioned, we constructed 5 different datasets from randomly sampling the original labelled set. We then trained 5 models using each dataset. We aggregated the predictions from all 5 models by choosing the label for each image with the highest vote from all the models. This new list of predictions gave us the highest accuracy score on our validation and test set of 91%.

Discussion & Conclusion

The key takeaways from this project are that to be able to have a small bias requires a good model that works for the specific problem. As we learnt in class no one model performs well on all problems. Using AlexNet while decreasing the number of channels in the convolutional layers, we were able to have a high accuracy on training set. The other key takeaway is that bootstrap aggregation can significantly decrease the variance of a model (increase in test accuracy of 5% on public set). Although our preprocessing methods and attempt to label the unlabelled dataset did not improve the accuracy, we would keep these two methods in mind if we were to continue the project. We feel that the way we used the unlabelled dataset was unique and successfully provided us with 50% (of original training set) more data that have high confident labels. Since this did help our solution, we believe that the lack of data was not a problem for this model. Moving forward we could investigate training a deeper model since we now have more labelled data. Alternatively, we could look towards other semi supervised learning techniques such as *Expectation Maximization* [2] or *Label Propagation* [3].

Statement of Contributions

Throughout this third mini project, all members contributed a significant and equal amount of time to complete all allocated tasks. Specifically, Hassan researched efficient preprocessing pipelines using torchvision and made sure our notebooks and files were structured and organized. Then, Anthony contributed notably with his flexible bagging implementation and Felix setup some network architecture and code to make the development of our solution efficient. Finally, all members contributed to the report.

References

[1] *Pytorch*. PyTorch. (n.d.). Retrieved November 21, 2021, from https://pytorch.org/hub/pytorch_vision_alexnet/.

[2] Wikimedia Foundation. (2021, November 20). *Expectation–maximization algorithm*. Wikipedia. Retrieved November 24, 2021, from https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm.

[3] *1.14. semi-supervised learning*. scikit. (n.d.). Retrieved November 24, 2021, from https://scikit-learn.org/stable/modules/semi_supervised.html?fbclid=IwAR2AmtZ7465BlRWIjme8lHl0QmS41_NPEtrsqZLV0pFJedWPHaNjOQP8IwU#label-propagation.

Appendix

GitHub repository: https://github.com/felixsimard/comp551-p3

Note: the repository visibility might be set to *private* when we submitted the assignment for academic integrity reasons.

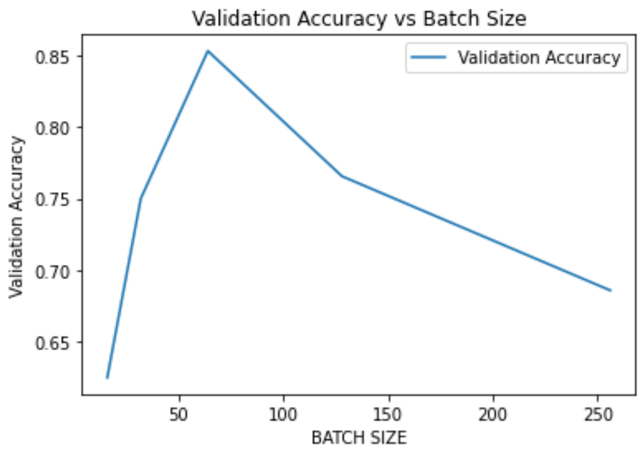

Figure 1: Validation accuracy versus batch size