COMP 551 Project 3 Report

Group 28
Ian Benlolo: 260744397, Christos Panaritis: 260739936, Tristan Shoemaker: 260640998
McGill University

March 6, 2021

Abstract

Traditional machine learning methods have difficulty matching human performance on image based classification tasks. Neural Networks (NN), and their expansions Convolutional Neural Networks (CNN) are have emerged as the standard method in machine learning for image classification. Our goal in this project is to classify images that are based on the MNIST handwritten number dataset, with modified backgrounds. Each image has three handwritten MNIST numbers, and the highest number is reported for each image. We compare two advanced CNNs that are well known top performers on the ImageNet image classification task: InceptionV3 and ResNet152V2. We have applied them to this classification task, with InceptionV3 giving a validation set accuracy of 97.3 %, and ResNet152V2 giving a validation set accuracy of 96.8 %, both performing very well. A "base" model was also implemented and tested which resulted in 86.8% accuracy which a much smaller training time.

Introduction

In the goal of advancing artificial intelligence (AI), much progress has been made in image recognition over the past decade. Whether in the goal of furthering autonomous vehicles or medical imaging, there have been many breakthroughs and benchmark datasets in machine learning and AI in general.

Supervised classification is a general two-step approach for classifying new data after having trained a model on classified data. In this, the analyst will "supervise" the classification of the problem in question. Many supervised learning algorithms exist such as SVMs, neural nets, logistic regression, naive bayes, random forests etc. These all use different approaches but have the same end-goal of classifying new, unseen data correctly. Convolutional neural networks (CNNs) have quickly become one of the most widely used techniques for image classification, otherwise known as computer vision. These are made up of "neurons", loosely analogous to biological neural systems, that have learnable weights and biases. Their success can mainly be attributed to advancements in GPU technologies and the emergence of large datasets [5].

The Modified National Institute of Standards and Technology database (MNIST) consists of handwritten digits ranging from $0 - 9$ and is a commonly used dataset for training many image preprocessing models. The classification task in this project consists of a "modified" MNIST, which are images containing three MNIST numbers over various backgrounds in a 128×128 grayscale image. We are starting from two well-proven CNNs for image tasks: InceptionV3 and ResNet152V2, and applied them to the classification task at hand.

Related Work

In the 2015 ImageNet Large Scale Visual Recognition Competition (ILSVRC), ResNet152 was the winner with a 3.57 % error rate on the ImageNet test set. As models get deeper, taking the partial derivative of the error function with respect to the current weight can result in the multiplication of n very small or very large numbers in an n-layer network. Consequently the gradients can vanish or explode for deep networks, resulting is lower accuracy despite having greater depth. In order to have a stable deep learning network, ResNet adds a "shortcut" connection between groups of weight layers, as shown in Figure 1 (a). This shortcut is an identity mapping which adds the outputs of the layers to where they rejoin the network. This allows for layers to learn a "residual mapping". If the gradient vanishes/explodes for the weight layers, the identity map will still be able to transfer back to earlier layers, avoiding this problem. These shortcuts add no parameters, do not significantly influence training time, and allow for extremely deep networks (in this case 152 layers) that are still easily optimizable. As proven by its performance in ILSVRC and testing by He et al., this technique is highly effective in lowering error rates [3].

InceptionV3 was the runner up in the same 2015 ILSVRC competition with an error rate of 3.58%. A driving concept behind the development of InceptionV3 is the reduction of computational cost in comparison to previous simpler networks such as VGGNet. InceptionV3 is based on "Inception" building blocks, which include convolutions, average pooling, max pooling, concatenations, dropouts and dense layers. These blocks can be seen in the architecture diagram in Figure 1 (b). In order to gain in performance, there is a generous use of dimension reduction using asymmetrical convolution layers, which take advantage of the correlation between nearby parts of an image. In total InceptionV3 has ~ 25 million parameters in comparison with ResNet152V2 with ~ 60 million parameters [4].

Our exact learning task with the specific backgrounds is novel, however the MNIST character set is a well known machine learning task. Chen et al. [1] report that their proposed CNN performs better than a human, with an error rate of 0.18%, an accuracy of 99.8%. We expect that performance on just the MNIST dataset will be better than our performance however, as the digits are isolated, without background and without rotations. Additionally, we have 3 digits to identify in each image.

Dataset and Setup

The training dataset consists of $50,000$ 128×128 pixel images containing three MNIST digits, labeled only the largest of the three digits, and the test set consists of $10,000$ unlabeled images. An example of these images can be seen in the Original column of Figure 2 with some of the background removed in the Processed column, as described in the Figure 2 caption. We observed that this did improve conversion speed, by aiding in simplifying the problem. However depending on the background this technique is not perfect, as seen in 2 row two processed. The classifications were only integer values and these were set categorically (they were turned into an array of length 10 containing 0's everywhere and a 1 at the index which consisted the classification).

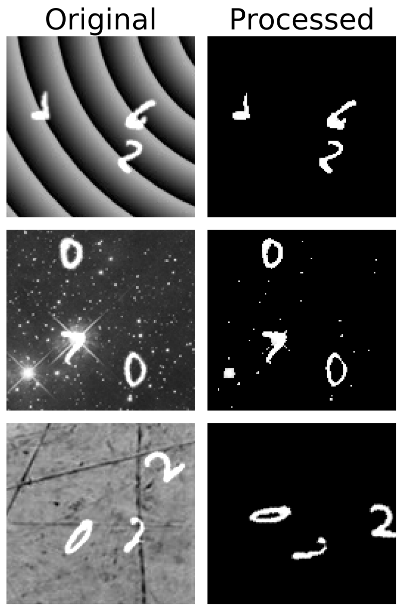

Figure 2: Example of background removal technique for three sample images. These are normalized to a maximum intensity of 1, and all pixels with a value below $1 - \frac{50}{255} \approx 0.8$ are set to 0, effectively removing the background. The third row additionally shows the effect of a rotation augmentation, in this case 45° clockwise.

All the models were run on Google Colab notebooks with GPU acceleration and the models/weights saved and downloaded for any post-processing/plotting. For

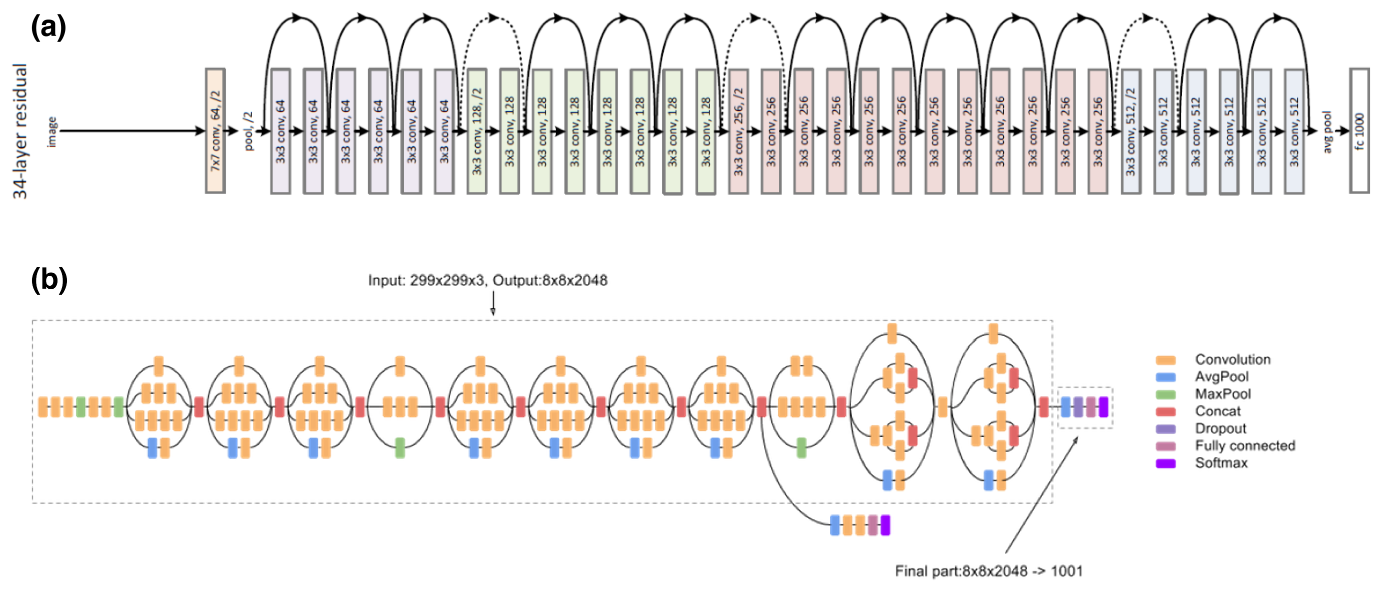

Figure 1: (a): ResNet152V2 architecture (b): InceptionV3 model architecture. The final output layer of both models are removed and both models are fed through a 2D polling layer and then three dense layers with 1024, 512 and 256 units respectively, with ReLu activation. This was then fed to a final 10 node Dense layer with Softmax activation for classification. *(a) is adapted from the Inception documentation: `https://cloud.google.com/tpu/docs/inception-v3-advanced`, (b) is adapted from [3].*

both models we are using Keras [2] with the TensorFlow backend.

Proposed Approach

We developed and applied the same training and evaluation regime for all three models in an effort to compare them. From the beginning, we made the assumption and plan that the logic for ”choosing the highest number” in each image would be learned into the CNN, without us explicitly defining it.

Models

As mentioned previously, the images were normalized to have a maximum value of 1, and the backgrounds were removed through the naive strict cutoff strategy. Both InceptionV3 and ResNet were imported pretrained on the ImageNet dataset in order to provide a strong starting point. In order to apply these models to our particular classification problem we added five additional layers at the end of each pre-trained model:

- 2D average global pooling layer: this is done to reduce the dimensionality of the vector before passing it to the dense layer.
- 1024-unit densely connected layer with ReLu activation
- 512-unit densely connected layer with ReLu activation
- 256-unit densely connected layer with ReLu activation
- 10-unit densely connected layer with SoftMax activation as the final classification layer.

These layers gradually reduce the dimensionality until our final classification layer fits output.

Additionally we developed a simple “Baseline” CNN model with 4 layers, each consisting of a convolution, a dropout and a max pooling layer for testing and benchmarking against the more complex models. The final layer is also a dense 10-unit layer.

Training

In order to take most advantage of the pre-trained models we used a multi-step training approach with an overall concept of gradually reducing generality. The Adam optimizer was used for all training segments, with categorical cross-entropy as the loss and categorical accuracy as the primary evaluation metric. The data was split into 85 % training and 15 % validation sets, and augmented "on the fly" as the images were used for training/validation.

Initially, all of the pre-trained model layers were frozen and only the additional layers were trained. Data augmentation for this step was a random rotation between 0 and 180 degrees, and a random background removal cut value between 20 and 70, allowing for the greatest variety of images. We then trained these layers for 15 epochs with a 256 batch size for both models.

All layers were then unfrozen and allowed to be trained. The maximum rotation range was reduced to 90 degrees to reduce the variance of input data, and then the whole model was trained for 50 epochs with a 256 batch size for InceptionV3 and a 64 batch size for ResNet152V2. This smaller batch size for ResNet152V2 was necessary to keep the VRAM usage within the limits of the available GPU with the larger number of parameters compared to InceptionV3.

Finally, we experimented with changing the batch size in the final training segment, finding that very large batch sizes, although slow, gave more stability to the training and allowed it to continue converging at high accuracy, resulting in a better final accuracy. This final training was run with a fixed background cut value of 70, and no rotation augmentation for 25 epochs with a batch size of 1024 for InceptionV3 and 128 for ResNet152V2.

The base model was run for 19 epochs with a batch size of 256 and no augmentation.

Results

We have also included the results of a simple "baseline" CNN for comparison with the more complex models. This model is much smaller (fewer parameters) and faster, but is unable to reach the high accuracies of the more complex models. As expected, our baseline model performed the worst with a validation set accuracy of 87.4 % and a Kaggle accuracy of 86.8 %.

For the ResNet and Inception models, after training them for a similar amount of epochs (around 90), the accuracy was 97.3% on the validation set and 97.1% on Kaggle for Inception and respectively 96.8% and 96.6% for ResNet (see Figure 1).

Since ResNet has many more parameters than Inception, it might take longer to converge and the convergence accuracy might be higher than Inceptions. However, looking at the loss graphs (Figures 4 and 3), it seems that both models have converged and thus any further training would be negligible (will not improve its accuracy). The loss graphs also indicate that ResNet had a much faster convergence than Inception. With these factors we can not say for certain that Inception is a better model than ResNet.

	Inception	ResNet	Baseline
1 Epoch TT [sec]	170	130	22
Evaluation Time [sec]	15.2	29.2	1.55
Val Accuracy [%]	97.3	96.8	87.4
Kaggle Accuracy [%]	97.1	96.6	86.8

Table 1: Comparison of time complexity and accuracy between the two tested models. TT: Training Time. Val: Validation Set. Training/evaluation times were run on Google Colab Tesla K80 GPUs. 1 Epoch training time was measured with a 256 batch size.

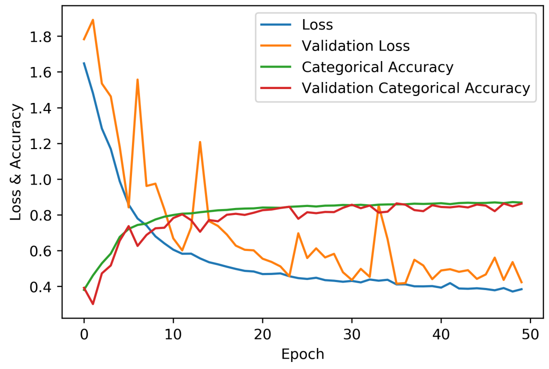

Figure 3: Inception loss and accuracy for training and validation datasets vs epoch. Initial training of the full InceptionV3 model + additional layers. As is expected, the validation set shows a much higher variance in both loss and accuracy between epochs in comparison to the training set, but it does follow the training set.

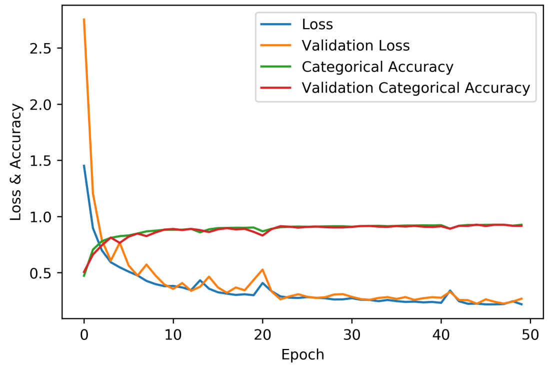

Figure 4: ResNet loss and accuracy for training and validation datasets vs epoch. Initial training of the full ResNet152V2 model + additional layers. There is a significant increase in loss around epoch 20. Inception does not exhibit features like this in its convergence.

In addition, due to the nature of the project, there is a computational limitation to the training. Simply looking at the evaluation run time (see Table 1), the Inception model is twice as fast as the ResNet model and the base model is about 5 times faster than Inception.

Thus, for this scenario, the results conclude that the Inception based model is better than the ResNet based model by a small margin (less then 1%). However, without the computation and time restrictions, it could very well be possible that the ResNet model would perform just as well if not better than the Inception based model.

Additionally, the training regime we used was developed through experimentation with Inception, and then applied to ResNet for comparison. It is possible that a different training technique would allow for better results from ResNet. These were not explored due to the time and resource constraints.

Discussion and Conclusion

Using state of the art models that have been proven to solve a similar problem allow for very quick development, and allow for a focus on the optimization of training, pre-processing and hyperparameters. Pre-trained models additionally reduce the necessary computing power required to effectively train a model, by starting it closer to convergence.

For this task, the results indicate that a modified InceptionV3 model was the best suited, achieving the highest accuracy of all the models tested. However as stated in the results the accuracy difference is small enough that changes in training or hyperparameters could result in ResNet achieving a higher accuracy. Both of these models are also probably more complex than strictly required for this task, having been developed for much more general image classification into many more categories.

It may be interesting to continue investigating different architectures to see if other models perform better. Another possible approach to the problem would also be dividing the classification problem into two parts: finding and isolating the digits, and then recognizing them. This technique could allow for the extremely high standard MNIST accuracy to hold for this modified task as well.

More work and time can also be put into training and testing simpler models since we showed that a simple four-layer model can achieve high accuracies of 88%, there may be models much simpler than InceptionV3 which attain accuracies in the high 90s.

Statements of Contribution

- Ian Benlolo worked on preprocessing, and on and creating, training and testing the baseline model as well as an equal contribution to the report.
- Christos Panaritis worked on creating, training and testing the ResNet based model as well as an equal contribution to the report.
- Tristan Shoemaker worked on creating, training and testing the Inception based model as well as an equal contribution to the report.

References

[1] Li Chen et al. "Beyond human recognition: A CNN-based framework for handwritten character recognition". In: *Proceedings - 3rd IAPR Asian Conference on Pattern Recognition, ACPR 2015*. 2016, pp. 695–699. ISBN: 9781479961009. DOI: `10.1109/ACPR.2015.7486592`.

[2] François Chollet et al. *Keras*. `https://github.com/fchollet/keras`. 2015.

[3] K He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: `10.1109/CVPR.2016.90`.

[4] Christian Szegedy et al. *Rethinking the Inception Architecture for Computer Vision*. 2016. DOI: `10.1109/CVPR.2016.308`. arXiv: `1512.00567`.

[5] Siham Tabik et al. "A snapshot of image pre-processing for convolutional neural networks : case study of MNIST". eng. In: *INTERNATIONAL JOURNAL OF COMPUTATIONAL INTELLIGENCE SYSTEMS* 10.1 (2017), pp. 555–568. ISSN: 1875-6891. URL: `http://dx.doi.org/10.2991/ijcis.2017.10.1.38`.