INT2 GROUP 24 NEURAL NETWORK

Group 24: George Tassou, Henry Overton, Dragos Stoican, Jacob Turner, Yu Li, I-Sheng Tsai

***Abstract*—We have been tasked with creating a neural network that is capable of identifying objects present in images given to us (the CIFAR-10 test set). The neural network has been designed using PyTorch. Throughout this report we have detailed our methods of creating such a neural network, explaining each step and the method taken to develop our final version of the network, as well as showing our evaluation of different techniques trialed through the process and showing diagrams of how our end model functions. The achieved classification error was 6.4%.**

I. Introduction

Previous convolutional neural networks have been applied to the task of image classification. Our task was to take these past examples and apply them through various experiments, to create a model that achieves the highest accuracy on the chosen data set. An early paper written by Yann LeCun in *Gradient-Based Learning*[1] showed that gradient based models could outperform older models reliant on human defined heuristics. This process of back-propagation trained the model with respect to a performance measure. As a result human defined heuristics are now rarely used. A more modern approach made by Agarap in *Deep Learning using ReLu*[2] used ReLu for the activation function in each hidden layer as well as replacing the softmax classification function. The last layer now being an activation function instead of a discrete probability distribution. The value of image classification to modern society is immense. Systems in the future will come to rely on the ability to recognise objects with potential life threatening implications in extreme scenarios.

II. Method

A. Network Structure

We used a convolutional neural network (CNN) with 4 VGG blocks, as mentioned by *Karen Simonyan Andrew Zisserman* [3], (repeated convolution, ReLU layer pairs) followed by a MaxPool layer and a dropout layer after each block. One addition we made to the VGG blocks was adding a batch normalisation layer between the convolution and ReLU layers. The number of filters in each convolution layers doubles with each block, starting at 64 filters in block 1 and ending at 512 filters in block 4. The first 3 blocks had a batch norm of 3 convolution ReLU layers, however the 4th block had 4 of such sequences. After all the blocks, we reduced the pooled feature maps down to the 10 output nodes with a flatten, a fully connected layer, a one dimensional batch norm, a dropout and one final fully connected layer.

B. Training Parameters

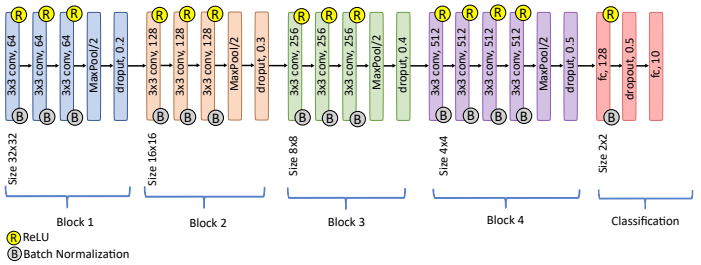
We applied an adaptive learning rate, based on the paper by *D.C. Cireşan et. al* [4], decreasing slightly each epoch, which led to a more successful training algorithm. By reducing the learning rate each epoch, the model plateaued after a greater number of epochs and with a lower error. We used the Adam optimizer with CrossEntropyLoss as our loss function, which also applies SoftMax to the data.

C. Training Process

We chose to augment our data by randomly applying one of the following transformations mentioned in [4] and *The Quest of Higher Accuracy for CNN Models* [5]: horizontal flip, Gaussian blur, or translation to counteract overfitting due to a relatively small data size.

To train our model we created a prediction for each image (In batches) then applied back propagation. We first zero the gradient to ensure the predictions of the previous batch do not affect the optimization of the current one. Then calculate the loss, comparing the models predictions with the correct labels. In the PyTorch implementation, the loss function applies a gradient to all the tensors in the network which gets updated by the optimizer to change the weighting of the network.

III. Model Diagram



IV. Results and Evaluation

A. Experimenting approach

We performed a number of experiments to evaluate the efficiency of our neural network by varying the hyperparameters whilst applying different machine learning techniques. These experiments were compared with one another using our evaluation metrics classification accuracy and logarithmic loss.

Each row of the tables represents a unique configuration of hyperparameters and techniques with their respective accuracy. Every experiment alters a single hyperparameter so we can accurately determine whether the change had any impact.

Effective configurations are used during the next experiments, with some new variation to test new hyperparameters and methods. If the changes seem to be ineffective they are not carried forward to the next experiments model configuration.

Small scale preliminary experiments were conducted to ascertain as to which hyperparameters and methods we should be varying during our experiments. It was discovered that

the number of epochs, batch size and learning rate should remain constant throughout our experiments as they have negligible effect on the accuracy and such allowing for great experimentation of other hyperparameters and methods. As such every experiment was done on 300 epochs, a batch size of 64 and a learning rate of 0.001.

EX	AR	OA	DA	LR	BN	DR	ET	T/E	ACC	LL
E1	3 conv, 3 pool, 2 linear	SGD	None	0.001	No	No	No	38	74	9.8E-2
E2	3 conv, 3 pool, 3 linear	SGD	None	0.001	No	No	No	34	73	2.7E-2
E3	13 conv, 4 pool, 2 linear	SGD	None	0.001	Yes	No	No	36	76	7.4E-3
E4	13 conv, 4 pool, 2 linear	Adam	None	0.001	Yes	No	No	61	79	7.6E-3
E5	13 conv, 4 pool, 2 linear	Adam	None	0.001	Yes	No	Yes	58	88	1.1E-3
E6	13 conv, 4 pool, 2 linear	Adam	translate, h flips, blur	0.001	Yes	No	No	51	62	1.4E-3
E7	13 conv, 4 pool, 2 linear	Adam	translate, h flips, blur	0.001	Yes	No	No	59	92	1.6E-4
E8	13 conv, 4 pool, 2 linear	Adam	translate, h flips, blur, v flips, rotations	0.001	Yes	Yes	No	59	91	2E-3
E9	13 conv, 4 pool, 2 linear	Adam	translate, h flips, blur	0.001	Yes	Yes	No	60	93	1.2E-4
E10	13 conv, 4 pool, 2 linear	Adam	translate, h flips, blur	0.001 adaptive	Yes	Yes	No	60	93.6	1E-4

EX - Experiment number, AR - Arhitecture, OA - Optimization algorithm, DA - Data Augmentation, LR- Learning Rate, BN - Batch Normalization, DR - Dropouts, ET - Error training T/E - Time per epoch, ACC - Accuracy, LL - Log Loss

B. Explaining results

The first major discovery occurred during E3 where the use of batch normalisation increased our accuracy by 2%.

After the first four experiments, we discovered a highly optimised architecture that was consistently reaching close to 80% accuracy. We then kept the architecture constant for the remaining experiments. During E2 we discovered that adding more than two fully connected layers was at the detriment to accuracy. During E3 we experimented with the number of convolutional layers as well as the number of number of channels within each layer. A modern design for this, as described in [3], where after each set of convolution layers the number of channels increases. This was a good improvement in accuracy, but the training time also increased drastically.

We then started altering the optimization algorithm in experiment E5, and based on *State-of-the-Art CNN Optimizer*[6] we discovered that Adam is best suited for image classification. This optimization algorithm takes advantage of adaptive learning rates to find individual learning rates for each parameter *Adam — deep learning optimization*[7]. This increased the accuracy by almost 10%.

Based on [5], we thought error training would be an effective technique to increase the performance, however it turned out to be detrimental for the accuracy and for the training time of the network, as observed in E6.

The next technique we applied was data augmentation. We first experimented with the suggested transformations in [4] and [5], and those techniques proved effective (E7) This technique resulted in an additional 4% accuracy. However some additional transformations, like vertical flips and rotations (E8) had a negative effect on the accuracy.

At this point the network was consistently reaching a point when it was massively overfitting. To deal with this issue we added dropout layers like described in *Dropout in PyTorch*[8]. This addition in E9 partially fixed the overfitting problem, helping us reach a new best accuracy of 93%.

The last experiment added an adaptive learning rate mentioned in [4] that further improved the model. The learning rate starts at 0.001 and it is scaled by 0.993 after every epoch.

After the final model reached an accuracy of 93.6%, it was decided to conclude experimenting as a high accuracy was reached. The model was trained for 5 hours on a 2080 Super GPU and a Ryzen 3800X using PyTorch 1.8 with CUDA 11.

V. Conclusion and Further Work

In conclusion, we have been able to develop a neural network capable of detecting objects with a relatively low error while only training for 5 hours (300 epochs), this is a good level of performance. Based on existing research and our low error, the choice of architecture can also be considered effective. During the project, we have managed to succeeded in applying a basic level neural network to a high accuracy by using effective image classification techniques. One problem we encountered was an error in our testing function, leading to a reduced accuracy due to testing effecting training.

In the future, we could implement more advanced architectures to further increase the accuracy of the network, for example inception modules could be used to achieve this.

References

[1] Yann LeCun & Leon Bottou & Yoshua Bengio & Patrick Haner *Gradient-Based Learning Applied to Document Recognition* Nov 1998 [Online]. Available: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

[2] Abien Fred M. Agarap *Deep Learning using Rectified Linear Units(ReLu* 7 Feb 2019, [ONLINE], Available https://www.researchgate.net/publication/323956667_Deep_Learning_using_Rectified_Linear_Units_ReLU

[3] Karen Simonyan & Andrew Zisserman, *VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION* 10 April 2015, [ONLINE]. Available https://arxiv.org/pdf/1409.1556v6.pdf

[4] D.C. Cireşan et. al, *Flexible, High Performance Convolutional Neural Networks for Image Classification*, 2010. [Online]. Available: https://people.idsia.ch//%7Ejuergen/ijcai2011.pdf

[5] S. Mhalagi, *The Quest of Higher Accuracy for CNN Models*, 2019. [Online]. Available: https://towardsdatascience.com/the-quest-of-higher-accuracy-for-cnn-models-42df5d731faf

[6] Yaqub M, Jinchao F, Zia MS, et al. *State-of-the-Art CNN Optimizer for Brain Tumor Segmentation in Magnetic Resonance Images.* Brain Sci. 2020;10(7):427. Published 2020 Jul 3. doi:10.3390/brainsci10070427 [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7407771/#:~:text=The%20Adam%20optimizer%20had%20the,ability%20in%20classification%20and%20segmentation.

[7] Vitaly Bushaev, *Adam — latest trends in deep learning optimization* 22 October 2018, [Online]. Available: https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c

[8] Ayush Thakur, *Dropout in PyTorch – An Example*, [Online]. Available: https://wandb.ai/authors/ayusht/reports/Dropout-in-PyTorch-An-Example--VmlldzoxNTgwOTE