Intelligent Systems 2: Neural Network Image Classifier

Group 25: Ayman Zahir, Adam Kirby-Stewart, Adam Johnson, Alexander Davis, Sam Laljee

***Abstract*—We were tasked with creating a Convolutional Neural Network to identify images in the CIFAR-10 Data set, and classify them into categories (”Car”, ”Bird” etc). To accomplish this we created a neural network that is loosely based on principles followed by the Alex-Net and VGGNet networks, although only inspiration was taken from them. This method of image classification achieved a maximum accuracy of 90.1% and an average accuracy of 89%. Hence, our CNN achieves a classification error of 11%.**

I. Introduction

OUR project required us to research, construct, train and evaluate a CNN from scratch for the purposes of 2D image classification against the predefined CIFAR-10 data-set [4]. We consequently tested the trained model on the same data-set and evaluated its performance. Additionally, we were tasked with improving the model while conducting research on image classification neural networks with the goal of minimizing the classification error to an acceptable level.

Within the wider field of image classification CNN architectures, sequential application of techniques such as convolutions, ReLU and max-pooling are commonplace and generally effective when constructing networks. Our base network initially consisted of a randomized combination of these and then improved by drawing inspiration from AlexNet [1], VGGNet [3] and further experimentation.

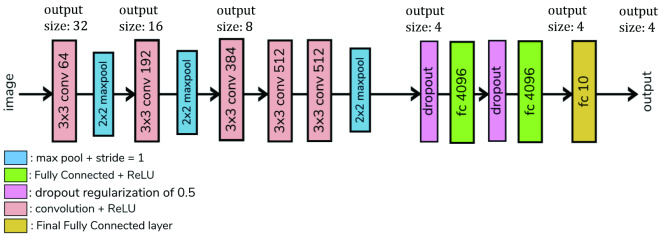
The Image Net challenge has been the main benchmark for Image Classification research since 2010. Image classification forms the basis for most other real time computer vision problems in various fields such as medical imaging. Improvements in this field directly tackle real world computer vision problems.

II. Methods

Our network features inspiration from AlexNet [1] and VGGNet [3]. As per Fig. 1, during training, the input to our network is a 32 x 32 RGB image. The images were preprocessed via data augmentation and normalization. The image is passed into the network; a stack of five 3x3 convolution layers with gradually increasing out-channels, with ReLU operations applied to each of them to remove any negative values [1]. The conv. padding is set to 1 pixel to preserve the spatial resolution. Spatial pooling is carried out by three max-pooling layers which follow some of the conv. layers and are performed over a 2 x 2 pixel window. We also used three fully connected layers. In the first two fully connected layers we also use dropout layers to zero a random element to prevent co-adaptation of neurons and avoid over-fitting. The final fully connected softmax layer has 10 channels to separate the images into the 10 CIFAR-10 categories.

Ideas from VGGNet can be seen in the final three serial conv. layers and in our use of solely 3x3 kernels for the convolutions to leverage a minimal receptive field [3]. We use 3x3 kernels, proving more effective than larger sizes [6].

Fig. 1. Network architecture



To calculate our loss function we used cross-entropy. Cross-entropy measures the performance of models which output probabilities between 0 and 1; this applies to our model as there is a choice as to which category the image matches and the cross-entropy can be calculated for each choice in our model, creating this equation for each image:

$$-\sum_{c=1}^{M} y_{o,c} log(p_{o,c}) \tag{1}$$

M = number of categories
y = binary indicator (0 or 1)
log = the natural log
p = predicted probability observation o is for class c

As for the training we used the SGD-optimiser with a learning rate of 0.001 and momentum set to 0.9. When training we iterated through images in the training set. The neural network would predict an image's label. The loss function is calculated for that image by comparing the predicted and actual labels. We use the loss to create new gradients for inputs of the next image, to ensure there is no accumulation of gradients the previous values are zeroed before the next are added. This is then run for multiple epochs to train the neural network.

III. Results and Evaluation

While iterating through models, testing and ablation studies were carried out to identify the accuracy and loss that our CNN produced. Hence we could gauge the effectiveness of our architecture. Our first CNN elicited a loss of circa 40% based on several runs and epochs.

To minimize this loss we applied a more structured architecture where additional convolution layers were introduced in conjunction with activation and linear functions, inspired by the AlexNet and VGGNet. This increased the activation of particular features within the image classes creating greater accuracy in predictions by filtering the locations to find strengths of detected features. This activation, which achieved better fitting results, was implemented using the Rectified Linear function and a neuron using ReLU (Rectified Linear Unit activation) rather than alternatives such as sigmoidal functions.

It was deemed beneficial to create a function that produced an accuracy table which rated the correctness of predictions per class. This more complex network improved the results and an indicative example can be seen in TABLE I.

TABLE I
MULTIPLE LAYER CNN

Class	Accuracy (c. 100 Epochs)
Plane	75%
Car	85%
Bird	62.9%
Cat	53.6%
Deer	68.5%
Dog	64.1%
Frog	78.2%
Horse	73.5%
Ship	85.7%
Truck	81.4%

Overall CIFAR-10 Accuracy: 72%
Overall CIFAR-10 Average loss: 28%.

The accuracy table proved useful in performing better informed ablation studies. We iterated across various values for strides within the convolutions as well as various optimisers such as AgaGrad, Adam and SGD. While two strides is the general convention for max pooling as it tends to improve spatial locality, especially for VGGNet architectures [3], a stride of one provided better accuracy as more features are extracted per pixel.

In terms of optimisers, SGD performed the most accurately according to the ablation study in TABLE II.

TABLE II
OPTIMISER STUDY

Optimiser	Indicative Accuracy (c. 100 Epochs)
optimiserSGD	75.5%
optimiserAdam	74.4%
optimiserRMSprop	71.0%
optimiserAdaDelta	36.7%
optimiserAdaGrad	65.8%

Although optimisers such as AdaGrad performed better during earlier epochs they plateaued faster before reaching a suitable accuracy. Conversely, SGD and Adam learned slower but kept gradually improving, with the former of the two improving our accuracy the most.

Despite these modifications, TABLE 1 shows our CNN struggles to classify animals possibly due to intra-class variability. To tackle this we employed image augmentation techniques; we pad each image by 4 pixels, randomly crop by 32 x 32 pixels and perform horizontal flips with a 50% probability. By adding these random dynamic transformations the model sees slightly different images in each epoch of training. Hence the model generalises features better in more varied orientations and arrangements, thus tackling intra-class variability [5].

After training the model using the aforementioned data augmentation techniques we achieved a significant improvement in our prediction accuracy as shown in TABLE III, reducing the average loss by more than half. Classes have had their loss reduced without decreasing any accuracy gained in previous network iterations.

TABLE III
AUGMENTED MULTIPLE LAYER CNN

Class	Accuracy (c. 100 Epochs)
Plane	91.9%
Car	92.2%
Bird	76.5%
Cat	73.3%
Deer	89.5%
Dog	80.8%
Frog	91.6%
Horse	90.7%
Ship	93.7%
Truck	94.4%

Overall CIFAR-10 Accuracy: 87%
Overall CIFAR-10 Average loss: 13%

Hardware resource used was a GPU (Nvidia GeForce GTX 1080) executing code hosted on Google Colab locally within 15 hours for 1000 epochs, eliciting the final average classification test error of 11%.

IV. CONCLUSION

The CNN created achieves as acceptable accuracy of 89% which demonstrates that our network can be confidently considered accurate due to the acceptable statistical margin of error for classification (circa 10%). Additionally we saw continual improvement in the speed of the network in providing reasonable losses, earlier each time after improvements. However, it struggled to classify animals, so for further work we would prefer structuring the network based on ResNet instead of VGGNet as it is unnecessarily complex; as adding more layers increases training error [2].

REFERENCES

[1] A. Krizhevsky, I. Sutskever and G. Hinton, *ImageNet classification with deep convolutional neural networks, Communications of the ACM*, vol. 60, no. 6, pp. 84-90, 2017.

[2] K. He, X. Zhang, S. Ren and J. Sun, 2015. *Deep Residual Learning for Image Recognition* [online]. Available: https://arxiv.org/pdf/1512.03385.pdf

[3] K. Simonyan and A. Zisserman, 2015. *Very Deep Convolutional Networks For Large-Scale Image Recognition* [online]. Available: https://arxiv.org/pdf/1409.1556.pdf

[4] "CIFAR-10 and CIFAR-100 datasets", Cs.toronto.edu, 2022. [Online]. Available: https://www.cs.toronto.edu/ kriz/cifar.html

[5] S. Kumar, 2019. *Data Augmentation Increases Accuracy of your model — But how ?* Medium, 2022. [Online]. Available: https://medium.com/secure-and-private-ai-writing-challenge/data-augmentation-increases-accuracy-of-your-model-but-how-aa1913468722

[6] A. Ihare, 2020. *Significance of Kernel size* Medium, 2022. [Online]. Available: https://medium.com/analytics-vidhya/significance-of-kernel-size-200d769aecb1