Deep Learning Project 1

Hari Vidharth s4031180

Ashwin Vaidya s3911888

Abstract

In this project, we compare the different Deep Learning architectures for the task of image classification. For this, we used LeNet5, Alexnet, VGG19, and Resnet50 architectures on the Intel image classification dataset. We compare and benchmark the models with different activation functions, optimizers, padding and regularization techniques such as dropout and batch normalization. We obtained the best results of 90% for the testing accuracy on VGG network pre-trained on *ImageNet* dataset.

1 Introduction

A common application in Computer Vision is to identify objects in a given scene. Earlier methods involved calculations to extract key points for matching images [7]. Other methods involved using functional transforms to detect faces [11]. However, these methods were superseded by Deep Learning methods when Alex et. al. [4] showed that Convolutional Neural Networks [6] can outperform all existing methods for the ImageNet dataset [2]. Following this, many such models were designed which increasingly performed better on this dataset. Karen et. al. [9] improved performance by designing deeper networks and Kaiming et. al. able to scaled it further by introducing skip connections [3].

In our project, we experiment with various architectures and present the configurations which give the best results. We experiment with various optimizers, activation functions and regularizers. Our results are presented in Section 3.

2 Experiment Design

2.1 Dataset

We used the Intel image classification dataset [10] to benchmark our models. It consists of natural scene images from around the world. It has 14000 training images and 3000 testing images. For some of the experiments, a subset of the training set was reserved for validation. Each image has a dimension of 150x150x3 and is categorized into six categories — buildings, forest, glacier, mountain, sea, street.

2.2 Architectures

For our experiments, we used Keras [1] to build and develop all the models and compare their performance on the dataset.

2.2.1 LeNet5 and AlexNet

Both LeNet5 and AlexNet model [5] were developed from scratch. The experiment was first conducted on the dataset without padding and batch normalization. In the subsequent experiments zero padding and batch normalization were introduced and tested. These models were tested extensively with different optimizers and activation functions while keeping categorical-cross-entropy as the loss function. The optimizers used for the experiments are SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam and the activation functions used are linear, exponential, sigmoid, hard sigmoid, tanh, ReLU, softsign,

softplus, SELU, softmax, ELU. Each combination was run for 100 epochs, and early stopping was used as a means for regularization to prevent overfitting by monitoring validation loss and keeping the patience to 10. The best results for all the combinations and experiments are presented in Section 3 and discussed in Section 4.

2.2.2 VGG19 and ResNet50

Both VGG [9] and ResNet50 [3] architectures were obtained from the Keras library. Each of the dense layer were followed by a Batch Normalization layer and dropouts of 0.2 as a form of regularization to prevent overfitting. The Adam optimizer was used for training the weights and categorical cross-entropy was used to calculate losses. We also compared the results of VGG by using a pre-trained network on *ImageNet* and replacing the fully-connected layers. The comparison between the results are tabulated in Table 2 and Table 3. Seeing the better performance of transfer learning in the VGG experiments, ResNet50 architecture was implemented using transfer learning in a similar manner. The top two results are presented in Table 3.

3 Results

Architecture	Optimizer	Activation Function	Training Accuracy	Test Accuracy	Padding	Batch Normalization
LeNet5	Adagrad	Relu	0.856	0.751	False	False
	Adam	Relu	0.992	0.725	False	False
	Adagrad	Relu	0.849	0.727	True	False
	RMSprop	Relu	0.994	0.713	True	False
	RMSprop	Relu	0.987	0.734	True	True
	SGD	Relu	0.999	0.731	True	True
AlexNet	SGD	Softsign	0.888	0.787	False	False
	Adagrad	Relu	0.966	0.781	False	False
	SGD	Softsign	0.940	0.8155	True	False
	Adagrad	Softsign	0.950	0.814	True	False
	SGD	Elu	0.999	0.842	True	True
	Adagrad	Tanh	0.999	0.831	True	True

Table 1: The table represents the top 2 best results for LeNet5 and AlexNet architectures with respect to Padding and Batch Normalization in terms of optimizer, activation function, training and testing accuracies.

The results for LeNet5 in Table 1, shows that the ReLU activation function gives the best result among all the other activations and for the varying combinations of optimizers, presence of padding and batch normalization we see similar results with the testing accuracies very close to each other while batch normalization slightly improves upon the training accuracy.

The results for AlexNet in Table 1, shows that in the model without padding and batch normalization, the two combinations of optimizer and activation that perform the best are SGD and Softsign, Adagrad and Relu. The combination of Adagrad and ReLU have a high training accuracy and a comparatively lower test accuracy while the combination of SGD and Softsign have closer training and testing accuracy. When padding is introduced the combination of Adagrad and Softsign performs better replacing the combination of Adagrad and ReLU in the top 2, while the combination of SGD and Softsign remains on top. In this case, both the combinations have similar training and testing accuracies and the performance of SGD and Softsign combination has been improved. When batch normalization is introduced the SGD and Elu combination performs the best and the training while the accuracies for Adagrad and Tanh combination are not far behind showing us the overall model performance is improved further.

The result of our VGG classifier with transfer learning is presented in Table 3. It is evident that PReLU activation function gives the best result. Also, it gives the highest testing accuracy.

When it came to ResNet, the Sigmoid activation gave the highest training accuracy while Elu had the best testing accuracy. From our experiments we conclude that transfer learning with Resnet gives

Activation Function	Optimizer	Training Accuracy	Testing Accuracy
ELU	Adam	0.609	0.608
Sigmoid	Adam	0.792	0.770

Table 2: Top Two Training and Testing Accuracy on VGG model without using Transfer Learning

the highest accuracy with the least loss when we use the ReLU activation function.

Model	Activation Function	Optimizer	Training Accuracy	Testing Accuracy
ResNet	ELU	Adam	0.911	0.880
ResNet	Sigmoid	Adam	0.914	0.877
VGG	ELU	Adam	0.990	0.880
VGG	PReLU	Adam	0.999	0.902

Table 3: Top two Training and Testing Accuracy on ResNet and VGG model using Transfer Learning

The results from the Table 2 shows that VGG when trained through all the layers gives the best results with the softmax activation.

4 Discussion and Conclusion

Table 1, shows us that adding zero padding to the AlexNet architecture improves the model performance, this is because padding helps in controlling the size of the input and output image providing more training points and also helps in preserving the edge data giving more details for training the image. Batch Normalization further improves upon the model performance by increasing the training and test accuracy and also reducing training time, this is because it standardizes the input and output between the layers, stabilizing the learning process and significantly reducing the number of training steps required to train the model. Also, it removes noise by subtracting the mean during the standardization process further helps in reducing overfitting.

However, according to Table 1, for LeNet5 architecture the zero-padding and batch normalization does not seem to affect the model performance and we get similar test accuracies. We beleive that this is because of the low complexity of the model compared to the classification task at hand and the complexity of the dataset.

The above experiments for LeNet5 and AlexNet were conducted with Early Stopping as a means to prevent overfitting. Too many epochs may lead to an overfit model and too less might underfit the data. Early stopping helps in streamlining the learning process by constantly monitoring a given metric, in our case the validation loss, with the patience of 10. This means that the validation loss will be monitored for 10 epochs and if the validation loss does not decrease after 10 epochs the training process is stopped thus reducing the training time and help control overfitting. We chose the value as 10 because initial experimentation with patience 5 yielded poor results.

A comparison between Table 2 and 3 shows that when transfer learning is used, it gives higher training and testing accuracies. This is because the initial layers learn lower-level feature extraction and are usually similar for each task leading to faster training. Further, it has been shown that knowledge transfer between domains leads to better generalization [8]. In our case ImageNet dataset is made up of everyday objects whereas the Intel dataset is of different scenery.

Observing the advantage of using transfer learning, our Resnet model was only implemented using transfer learning with varying activations. Our experiment shows that for ResNet, sigmoid activation gave the best performance.

We have used the Testing accuracy as our metric for comparison. When comparing and benchmarking different architectures, accuracy is a very good metric. The ImageNet competition uses this accuracy metric to compare and rank the different models. Also, the dataset creator mentioned the need to develop a deep learning model with the best accuracy for this data [10].

From the results obtained we conclude that VGG19 with transfer learning, dropout of 0.2 and batch normalization gave the best performance.

References

[1] François Chollet et al. Keras. https://keras.io, 2015.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[5] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[6] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999.

[7] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[8] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359, 2010.

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[10] Analytics Vidhya. Intel image classification, 2019. data retrieved from Kaggle, https://www.kaggle.com/puneet6060/intel-image-classification/version/2.

[11] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001.