Deep Learning Fundamentals - Assignment 2

Gauranga Das
University of Adelaide
a1910652@adelaide.edu.au

1. Introduction

Convolutional neural networks (CNNs) [7] have recently gained popularity in computer vision for tasks such as image classification and object detection. Compared to other neural networks, CNNs are capable of modeling complex relationships in images by encoding spatial information through convolutional kernels. They have outperformed traditional computer vision methods in a variety of applications.

In this project, the task is to perform image classification on a dataset using different CNN architectures. The main objective is to explore various CNN architectures, perform hyperparameter optimization, calculate performance metrics, and compare their effectiveness. This project will evaluate the strengths and weaknesses of these architectures and assess their suitability for real-world applications, suggesting potential improvements for future work. The Food-11 [1] dataset from Kaggle will be used for this analysis.

For this task, pre-trained models from PyTorch are utilized. Instead of training the entire model, most layers are frozen, and only a few of the final fully connected layers are fine-tuned.

2. Method

Our goal is to perform image classification classification on the Food-11 dataset using different CNN architectures and then compare their performance.

2.1. Baseline

The baseline model will be a small CNN architecture. It has three convolutional layers with kernels of size 3×3 and each convolutional layer is followed by a ReLU activation and max-pooling layer. Finally, there is a fully-connected layer with 512 neurons and another fully-connected layer with neurons equal to the number of classes. It will be trained using the Adam [5] optimizer and a learning-rate of 0.001. The reason for choosing this as a baseline model is that it is relatively simple to implement and small compared to other neural network architectures.

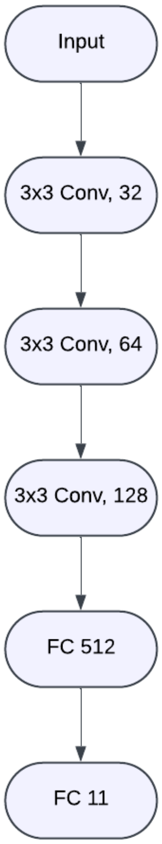

Figure 1. Baseline architecture

2.2. AlexNet

AlexNet [6] was one of the first large-scale CNN architectures which achieved excellent performance on the ImageNet dataset in the ISLVRC-2012 competition. The overall architecture consists of eight layers; the first five are convolutional layers and the final three are fully-connected layers. Response-normalization is applied to the output of first and second convolutional layers which normalizes the outputs to aid generalization. Max-pooling is applied to the outputs of response-normalization layers and fifth convolutional layer. ReLU activation function is applied to the output of each layer. The first convolutional layer uses kernels of size 11×11. The second convolutional layer uses kernels of size 5×5. The remaining convolutional layers use kernels of size 3×3. Finally, the fully-connected layers all have 4096 neurons except for the final output layer where number of neurons is equal to the number of possible classes. To reduce overfitting, dropout is used.

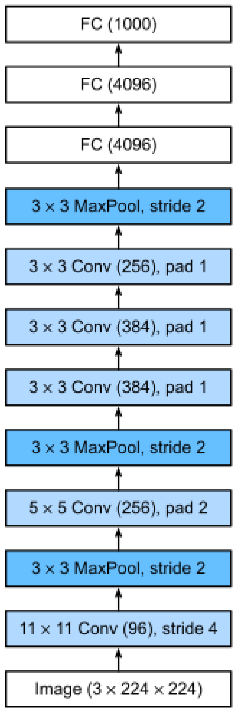

Figure 2. AlexNet architecture

2.3. VGG

The VGG [9] architecture improves upon AlexNet. The primary objective was to investigate the effect of convolutional network depth on image classification performance. The architecture has multiple 3×3 convolutional layers followed by max-pooling. The stack of convolutional and max-pooling layers is followed by three fully-connected layers with 4096 neurons. The final layer performs the classification using softmax which has number of neurons is equal to the number of possible classes. The ReLU activation function is applied after each convolutional layer and fully-connected layer. Various models with different depths are explored during experiments. Unlike AlexNet, where a convolutional layer with kernel of size 11×11 or 5×5 is followed by max-pooling, VGG-16 has mutliple convolutional layers with kernels of size 3×3 followed by max-pooling. Using multiple convolutional layers with smaller kernel size reduces the number of parameters required for training and introduces more non-linear activation functions. Also, compared to AlexNet, VGG-16 has greater depth due to a larger number of convolutional layers. The paper demonstrated that increasing the depth of CNN architectures can lead to significant improvement in performance.

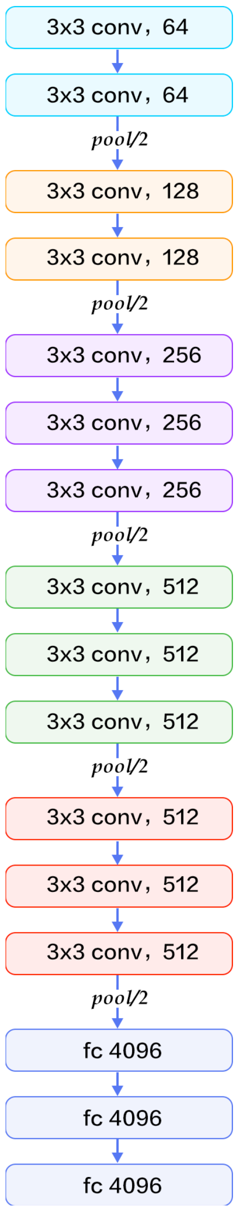

Figure 3. VGG-16 architecture

2.4. ResNet

Increasing the depth of neural networks enhances it's performance. However, after a certain point, stacking more layers causes the accuracy to saturate and then degrade rapidly. Very deep neural networks are therefore, harder to optimize. It is postulated in the paper that the reason for this could be that multiple stacked layers have difficulty in learning identity mappings. ResNet [3] rectifies this problem by introducing residual connections. With residual learning, it is much easier to learn the identity mappings since, the input is directly added to the output of stacked layers. Residual learning is defined as:

$$\mathbf{y} = F(\mathbf{x}) + \mathbf{x}$$

Here $\mathbf{x}$ is the input of the residual block, $F(\mathbf{x})$ is the output of the stack of convolutional layers and finally $\mathbf{y}$ is the output of the residual block which then passed as input to the next residual block.

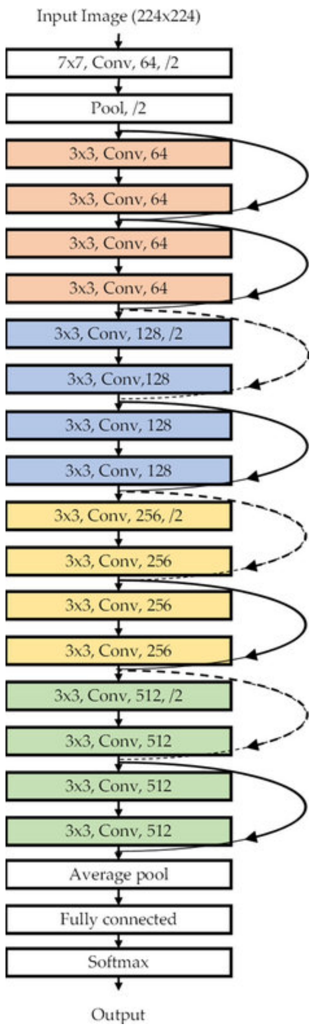

Figure 4. ResNet-18 architecture

2.5. Food-11 Dataset

For the experiments, the Food-11 dataset was chosen. This dataset contains images of different types of food. The dataset downloaded from Kaggle has been split into training, validation and test sets. The reason for choosing this dataset is that it is challenging to perform image-classification on such a dataset. Image classification generally becomes harder as the number of possible classes increase. In the case of Food-11 dataset, there are 11 different classes which can be difficult for a neural network. For training, each class has approximately 400-1500 images, which is low considering that deep learning typically requires thousands of images per class to perform well. Another challenging aspect is that the number of images for each class are not evenly distributed as shown below in the histogram of distribution of labels.

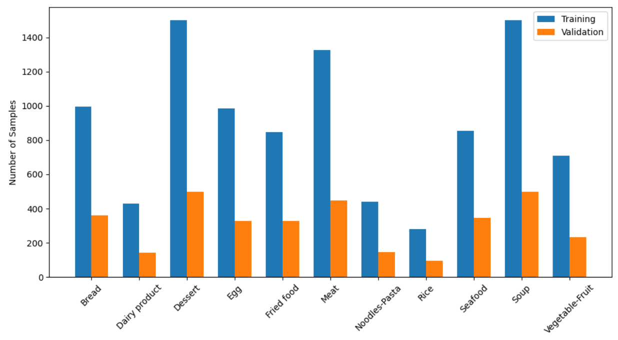

Figure 5. Distribution of labels in Food-11 dataset

These factors make image classification challenging on the Food-11 dataset.

3. Experiments

Different CNN architectures were trained and evaluated on the Food-11 dataset. The experiments illustrate that CNNs are very effective in performing image classification on image datasets.

Data Preprocessing. The Food-11 dataset is used from Kaggle. The dataset has three folders containing the training, validation and test sets. Each image is resized to 224×224. Using the TrivialAugmentWide [8] function from Pytorch, data augmentation is applied on the images. The function applies randomly applies an augmentation to each image such as rotation, color adjustment, shearing etc. Data augmentation can prevent overfitting and make the model more robust by making the model invariant to transformations such as rotation, shifting, change in brightness etc. Finally, each image is normalized using means of $0.485, 0.456, 0.406$ and standard deviations of $0.229, 0.224, 0.225$ respectively for each colour channel.

Hyperparameter Optimization. Four different CNN architectures- AlexNet, VGG-16, ResNet-18 and a baseline model are trained. To ensure each model performs at their best, hyperparameter optimization is done for each of them except for the baseline model. For each architecture, different combinations of optimizer and learning-rates are used to train a model and then evaluated on the validation set. Since, the pre-trained models takes a lot of time to train only four different hyperparameter combinations are tried. The Adam and RMSprop [4] optimizers and learning-rates of 0.001 and 0.0001 are used. Each combination of optimizer and learning-rate is used to train each architecture on the training set and then evaluated on the validation set to measure the performance of both the architecture and corresponding hyperparameters. The results on the validation set is used to obtain the best possible combination for each architecture.

Evaluation Methods. Different hyperparameter combinations are tried for each type of CNN architecture. For each architecture and it's corresponding hyperparameter combination, the model is trained on the training set and evaluated on the validation set for six epochs. The training and validation loss for each epoch was recorded. The losses were used to plot the loss curves to compare each hyperparameter combination. After training, the model was evaluated on the validation set to asses the performance of the model's corresponding hyperparameter combination. A confusion matrix is plotted to to display the model's per class performance. The overall accuracy, top-5 accuracy, balanced accuracy and per-class accuracies are also calculated for each architecture and it's corresponding hyperparameter combination. Accuracy alone is not a suitable metric to assess a model's performance since the dataset is imbalanced. Classes with large number of samples will impact the overall accuracy more than classes with less number of samples. Balanced accuracy is calculated as the average of recalls for each class. It addresses the imbalance problem by giving equal weight to the accuracy of each class. Therefore, the balanced accuracy is a more suitable metric. The architecture and hyperparameter combination with the highest balanced accuracy is chosen as the best model. Finally, the best model is evaluated on the test set.

Results. Table 1 compares the performance of various CNN architectures and their corresponding hyperparameters. It can be observed that the baseline model performs the worst with an overall accuracy of 32.54% and balanced accuracy of 29.50%. This implies that the model is too simple and is therefore underfitting. The overall accuracies for the other architectures range from 60-78% which is satisfactory performance. The top-5 accuracies on the other hand is consistently above 92%, which is excellent. This implies that even though the models were not able to consistently predict the correct class as it's top choice, the correct class was still among it's top-5 predictions which means that when a model made an incorrect prediction, it was still very close to predicting the correct class. This suggests that the models could be further improved by training for more epochs or trying different more hyperparameter combinations. Looking at the accuracies, it is clear that the VGG-16 architecture outperforms the AlexNet architecture by a significant margin. VGG-16 has more convolutional layers compared to AlexNet implying that increasing the depth of a neural network can lead to significant increase in performance. However, the ResNet-18 architecture performs worse than VGG-16, even though it has more convolutional layers and residual connections. One reason for this could be that larger neural networks also need more data to properly learn. Since, the size of the dataset is not very large, there isn't enough data for ResNet-18 to properly learn. This implies that increasing the depth or adding resid-

Table 1. Performance metrics (in %) on the validation dataset

Architecture	**Optimizer**	**Learning Rate**	**Accuracy**	**Top-5 Accuracy**	**Balanced Accuracy**
Baseline	Adam	0.001	32.54	81.43	29.50
AlexNet	Adam	0.001	65.10	94.78	66.94
AlexNet	Adam	0.0001	70.41	96.36	71.99
AlexNet	RMSprop	0.001	65.22	95.63	64.05
AlexNet	RMSprop	0.0001	69.65	96.18	69.32
VGG-16	Adam	0.001	73.85	96.33	74.73
VGG-16	Adam	0.0001	77.58	97.96	77.06
VGG-16	RMSprop	0.001	68.63	92.22	66.25
VGG-16	**RMSprop**	**0.0001**	**78.78**	**97.81**	**78.39**
ResNet-18	Adam	0.001	71.17	96.88	71.12
ResNet-18	Adam	0.0001	63.73	93.88	59.65
ResNet-18	RMSprop	0.001	69.18	96.53	68.20
ResNet-18	RMSprop	0.0001	63.03	94.46	59.50

Table 2. Performance metrics (in %) of the best model on the test dataset

Metric	**Result**
Overall Accuracy	80.37
Top-5 Accuracy	98.30
Balanced Accuracy Accuracy	80.21

ual connections does not always lead to an improvement in performance and can sometimes even cause degradation performance. The size of the neural network should be appropriately chosen depending upon the size of the dataset. Looking at the balanced accuracies, the VGG-16 architecture with RMSprop optimizer and learning-rate of 0.0001 performs the best since it achieves the highest balanced accuracy of 78.39%. Table 2 summarises the performance of the best model on the test set. It achieves an overall accuracy of 80.37% and balanced accuracy of 80.21%. It is able to achieve a decent performance but, it is not excellent. This suggests that the best model has room for further improvement. It's top-5 accuracy however, is 98.30% which is excellent.

4. Code

GITHUB LINK

5. Conclusions

In this paper, various CNN architectures were trained on the Food-11 dataset. The baseline model's overall accuracy and balanced accuracy were poor, but the other models performed significantly better. Pre-trained models are trained on large diverse datasets and have already learned useful features which helps them to converge faster on smaller datasets. This suggests that, when performing image classification on a small dataset, it is preferable to use pre-trained models rather than creating a CNN from scratch. Although the pre-trained models achieved decent overall and balanced accuracies, they were not exceptional, indicating that these models may not be suitable for real-life applications. However, the top-5 accuracies were excellent, implying that the models were often close to predicting the correct class, even when they made incorrect predictions.

Among the models tested, the VGG-16 architecture outperformed both AlexNet and ResNet-18. The findings suggest that increasing the depth of a neural network can improve performance, but only up to a certain point. Adding layers unnecessarily could even lead to a degradation in performance, especially on small datasets. Therefore, the depth of a neural network should be carefully chosen based on the dataset's size.

For further improvements, the CNN architectures could be trained for more epochs, and more hyperparameter combinations could be explored. Additionally, a learning rate schedule that reduces the learning rate after a certain number of epochs could be implemented. These optimizations could further enhance the model's performance and accuracy. Another alternative to CNNs could be Vision Transformers [2], which use attention mechanisms to perform image classification.

References

[1] Aleksandr Antonov. Food-11. `https://www.kaggle.com/datasets/trolukovich/food11-image-dataset`, 2019. Kaggle.

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[4] Geoffrey Hinton. Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude. `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`, 2012. Coursera: Neural Networks for Machine Learning.

[5] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

[7] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[8] Pytorch. TrivialAugmentWide. `https://pytorch.org/vision/main/generated/torchvision.transforms.TrivialAugmentWide.html`. (accessed 7 Oct. 2024).

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.