CNNs for Image Classification

Sankalp Pandurang Waghe
University of Adelaide,
Adelaide, South Australia, Australia
a1935053@adelaide.edu.au

Abstract

Computer vision faces a significant challenge, which is the task of image classification – because of applications like autonomous vehicles and facial recognition to be realized. Overriding this difficulty sees an attempt at data science, as the project is about classifying images from the CIFAR-10 dataset, which is a pool of various classes of color images in 32x32 pixels. So many applications our society relies on these days call for object recognition and when pictures are properly classified then total classification is achievable.

1. Introduction

One of the primary goals of image classification is self explanatory in itself that is to be able to categorize the images into certain classes and without them largely due to the variations that might appear within these images along the different brightnesses, angles, and backgrounds.

Convolutional Neural Networks (CNNs) are quite special in such realms because they are a type of deep learning model which is optimised for processing trodden data in places like pictures. The paper at hand specifically aims at constructing and training a Convolutional Neural Network on a well-known collection of a particular kind of data: the CIFAR-10 dataset.

CIFAR-10 is a dataset that consists of 60,000 32x32 colored images in ten categories, such as airplanes, automobiles, birds, and cats. Every class provides specific visual cues hence the model has to learn to differentiate multiple categories. Due to the low resolution of images and the distribution of many classes, CIFAR-10 is a non-trivial open set classification task in which models have to exhibit good generalization across different.

Comparision: Due to the long presence of the VGG and ResNet architectures in the deep learning world, it can be noticed that increasing the complexity of the model has its own drawbacks. As one of the fastest-growing fields, we decided to propose a relatively simple version for the problem of image classification, which preserves a good accuracy and is "light" like all the models before on the level of available computing resources.

2. Methodology

1. Model Architecture

Adding three Conv2D layers with filters of 32, 64, and 64 improves the model's ability to comprehend intricate details in images. MaxPooling2D layer decreases image dimensions, preserving key features and decreasing computational load. The final part of the model, a fully-connected Dense layer, has a 96% connection rate and 5% drop out rate.

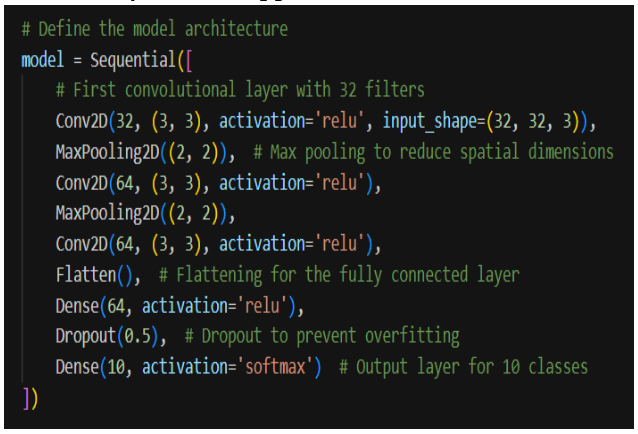
The building elements of these so-called CNNs include:

- Convolutional Layers: In three consecutive Conv2D layers with increasing steps of filters (32, 64, and 64), a filter shape of (3x3) is commonly prescribed as also ReLU activation function.
- Pooling Layers: One thing that often helps to simplify things is the ability to use MaxPooling2D (2x2) layer after every Conv2D layer.
- Flatten Layer: The last layer, which is also a fully connected layer, has this flattened layer that turns all 2D matrices into a 1D vector.
- Fully Connected Layers: A Dropout layer with a rate of 0.5 to diminish overfitting effects is placed after a Dense layer with 64 neurons and ReLU activation.
- Output layer: is a dense layer, with 10 neurons and softmax activation for multinomial class classification.

2. **Hyperparameters and Software tools**
    - Optimizer: Adam optimizer, was opted for because of its ease of use in terms of tuning parameters.
    - LossFunction:the multi-classification function was a categorical cross-entropy function.
    - epoch: 10
    - Batch size: 64
    - Software: TensorFlow and Keras libraries for model development, training, and evaluation.

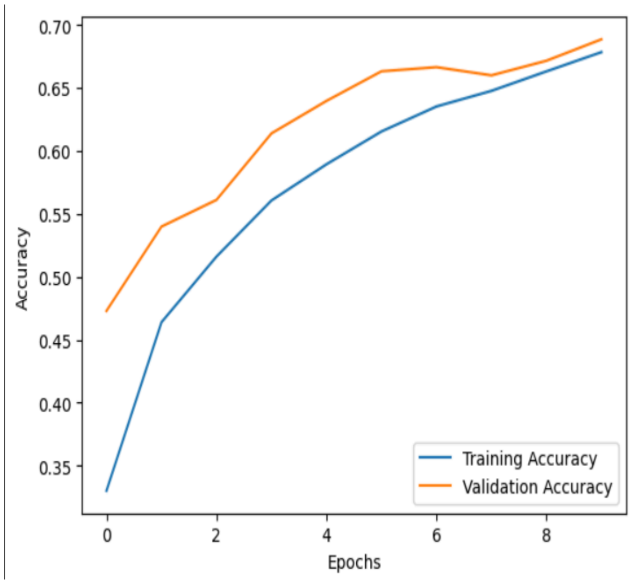
3. **Method Implementation**

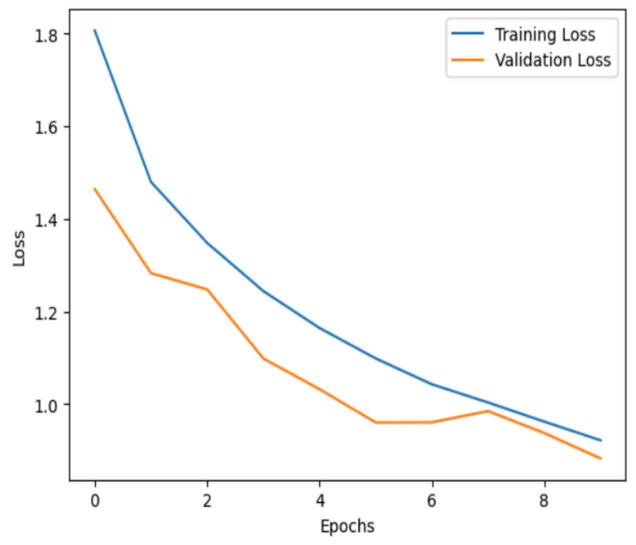
1. Code Summary
    - Data Preprocessing: As for the images in CIFAR-10, they are normalized while the labels are one-hot encoded.
    - Model Training: A 10-epoch training regimen shall be conducted for the purposes of assessing the training and overfitting of the model.
    - Evaluation: With the help of the accuracy metrics, the model performance will be evaluated on the test data and the results might be represented with accuracy/loss plots.
2. Key Code Snippet



```
# Define the model architecture
model = Sequential([
    # First convolutional layer with 32 filters
    Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    MaxPooling2D((2, 2)),  # Max pooling to reduce spatial dimensions
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    Flatten(),  # Flattening for the fully connected layer
    Dense(64, activation='relu'),
    Dropout(0.5),  # Dropout to prevent overfitting
    Dense(10, activation='softmax')  # Output layer for 10 classes
])
```

4. **Results**





1. **Training and Validation Accuracy:**

Over ten epochs, there are consistent gains in both training and validation accuracy, with both metrics closely convergent in the later epochs.

1. Training Accuracy: After the ten epochs, the training accuracy had risen from about 35% to roughly 69%.
2. Validation Accuracy: shows a similar upward trend, slightly outperforming the training accuracy throughout the epochs, peaking at around 70%.

This pattern was observed to established that the study model embedded the learnt information drawn from the train data without it overfitting much. Normally when accuracy is rising fast in validation compared to training with not much spread this means the model is learning well.
Although the generalization learning aspect can be said to have been achieved, its prediction is not very high p̃(70%). the CIFAR-IO dataset is more complex than the MNIST dataset and therefore improvement in the overall accuracy is not as difficult as reduction in the error rate per pixel.

4.2. **Training and Validation Loss:**

- Training Loss: Monitoring The training loss is practically reduced by the rate of 1.8 percent. Therefore it can be argued that the loss reduction of the training data with the progress of the model is quite effective – it decreased by almost 1.0 by the 10th epoch.

- Validation Loss: The loss of the validation cells decreased proportionally to that of the training loss with minor adjustments. We even noticed a small decline once validation loss started plateauing in the final days of the training. Validation error gradually approached below the training error, which indicates that there are possibly no major cases of the overfitting of systems.

In addition, the very nearness of the training-loss and validation-loss curves suggests that the challenge of balancing models between learning best and allowing for ignorance of some truth was perfectly met.

4.3 Test Set Performance

Upon performing tests, post-training revealed that the model scored an accuracy of 69.26%, on the test set. The figure falls within the same bracket as was observed during validation, that is, the model did indeed extend well to untested samples.

A successful rate of about 69% is acceptable for an unpretentious CNN model with a short training phase. Due to the simple nature of the dataset such as CIFAR-10, as it consists of a large array of image classes and different characteristics from their solution, it is necessary to build deeper networks or perform data augmentation to achieve higher results compared to the initial scores.

4.4 Discussion of Results

Several findings stand out from the operation of this CNN architecture:

- **Model Generalization:** the training, validation, and testing accuracies are very close to one another, giving an impression that the model has generalized well. Applying a dropout layer in the network may also have helped in lowering the chances of the model to overfit; more so because this is a relatively shallow neural network.
- **Accuracy Ceiling:** Despite it is not a royalty model yet based on the displayed results, it

can be argued that from figures accuracy tend to peak at approximately 70%. This could be interpreted as a signal to move on to the more complicated models or increase the size of the model further by the addition of more layers and or connections maybe residual connections whilst retraining more unfathomable data about CIFAR-10 families.

- **Room for Improvement:** In order to enhance the success rate of the model on the CIFAR-10 tests it is possible to consider other methods that could include data augmentation accessorizing the model, performing changes to the hyper parameters or applying pre-trained model (transfer learning).

5. **Reflection on Results**

The results produced so far sing out that even though the selected CNN system design was able to extract the needed information and transfer it to another problem, it would fail in extracting the minutiae contained in CIFAR-10 dataset. While exploring computer vision through image classification, this project also foresees the necessity of some improvements for any subsequent research.

6. Reflection on Project

6.1 Design Choices:

The important solutions included three compact dishes with CNN architectures to balance efficiency. The suppression through the cup-off helped to reduce reconstruction. The Adam optimizer was chosen because it has an adaptive learning rate feature that supports faster convergence.

6.2 Challenges and Solutions:

Challenges: Initially, overfitting was observed on the training data as the validation accuracy lagged behind the training accuracy

Solutions: Dropouts after a completely connected layer have reduced excess conformity by preventing the simultaneous installation of neurons.

6.3 **Future Work**

- Architectural Modifications: Improving the existing system: improving the accuracy of the algorithm on more complex datasets can be related to the increase of the number of convolutional layers and residual connections.
- Data Augmentation: techniques like rotation, flipping and scaling the data can help in enhancing dataset variability.
- Fine Tune Hyperparameters: Refining the learning rate, the allocation size and other hyperparameters could increase the test performance further.

7. Limitations

Even as this work successfully illustrated the applicability of Convolutional Neural Networks(CNNs) to categorize images. Such as in the context of the CIFAR-10 dataset, it is also pertinent to mention some constraints that provide input on aspects in which the model can be enhanced and the project can be widened with enough funds.

- Model Complexity and Depth:
  Our simple CNN model finds it challenging to pick up on patterns in CIFAR-10 images, especially next to models like ResNet and DenseNet. To work around this limitation, we tried adjusting it for better results and smoother experimentation.
- Data augmentation: Augmenting data is altering the repressive data set in machine learning so that the same analysis is done using that augmented data rather than doing the analysis only on original data. Techniques of this kind include image flipping, rotation, change of brightness, and the like, and improving the network's performance on previously unseen images

8. Conclusions

A CNN has been developed to work with the CIFAR-10 database and the results show that the model worked great on the provided CIFAR-10 data set. The selected architecture was found to be satisfactory, i.e. accuracy and computational requirements were taken into account.
The experiments were implemented very systematically in the course of the project and their results show the importance of various design decisions such as folding layers or adding drop out for better tuning of the CNN system. In other words, this study has already broken ground for further research in using deep learning to classify image.

References

- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems.
- Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. International Conference on Learning Representations.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. IEEE Conference on Computer Vision and Pattern Recognition.

11. **Appendix**

The code and data used in this study are available at https://github.com/sankalpWaghe24/CNN_ImageClassification.git