Deep Learning Project 2 Report

Olteanu Fabian Cristian

FMI, AI Master, Year 2

1. Introduction

The task at hand is to classify a set of images containing 150 different Pokemon. The dataset contains roughly 7000 samples. The methods that were used to perform this task were three different convolutional network architectures.

2. Dataset

The dataset is imbalanced, as shown in the histogram below (fig 1). As such, data augmentation was used to compensate this fact. For the training data generator, various augmentation techniques such as rotation, zooming, shifting, shearing, and flipping are applied to increase the diversity of training samples and improve the model's generalization. The testing data generator only applies rescaling to normalize pixel values.

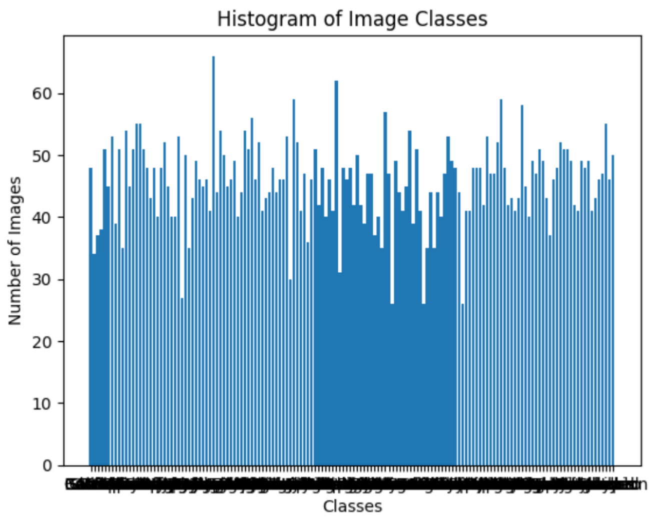

Figure 1. Dataset histogram

Before training any models using this dataset, it was split into three folds: train, validation and test using the following ratios: 0.75, 0.15 and 0.1.

3. VGG16 Implementation

VGG16 is a classic CNN architecture known for its simplicity and effectiveness. It consists of 13 convolutional layers and 3 fully connected layers. The architecture is characterized by stacking multiple 3x3 convolutional layers followed by max-pooling layers. Batch normalization is applied after each convolutional layer to stabilize and accelerate the training process. The VGG16 architecture is trained on the Pokemon dataset using stochastic gradient descent (SGD) optimizer with a learning rate of 0.001. The model is compiled with categorical cross-entropy loss and evaluated based on accuracy and F1-score metrics. In the case of training it from scratch on this dataset, however, it yielded very poor results (and so did AlexNet).

3.1. Comparison with the original paper

Comparing this architecture to the original[1], the most obvious change becomes apparent when looking at the added batch normalization layers, which were added after some initial tries when the model wouldn't learn. Additionally, the input layer was adapted to 64 by 64, which is different from the original VGG16 input layer, which accepted 224 by 224 images.

4. AlexNet Implemention

AlexNet[2] is a pioneering CNN architecture that significantly contributed to the advancement of deep learning in computer vision. It consists of eight layers, including five convolutional layers and three fully connected layers. The architecture incorporates overlapping max-pooling and dropout regularization to prevent overfitting. The AlexNet model for Pokemon classification is trained using SGD optimizer with a learning rate of 0.001. It is compiled with categorical cross-entropy loss and evaluated based on accuracy and F1-score.

Below, the original architecture is described, along with comparisons to the actual implementation in this project:

- AlexNet takes input images of size 227x227x3 (227 pixels in height and width, with three color channels for Red, Green, and Blue). In the implementation, this was changed to 64x64x3, as this was the size of the preprocessed images.
- AlexNet consists of five convolutional layers, denoted as Conv1 to Conv5. These layers extract features from the input images using learned filters.
- Conv1: 96 filters of size 11x11x3 with a stride of 4
- Conv2: 256 filters of size 5x5x48 (48 channels are the output from the first layer) with a stride of 1.
- Conv3: 384 filters of size 3x3x256 with a stride of 1.
- Conv4: 384 filters of size 3x3x192 with a stride of 1.
- Conv5: 256 filters of size 3x3x192 with a stride of 1.
- Note: The custom AlexNet implementation uses different filter sizes and strides for max-pooling compared to the original AlexNet. Additionally, the implementation in this project includes batch normalization layers after each convolutional layer, which one would have believed that it would help in stabilizing and accelerating the training process.
- ReLU (Rectified Linear Unit) activation functions are applied after each convolutional layer to introduce non-linearity.
- There are three max-pooling layers in AlexNet. These layers downsample the feature maps obtained from the convolutional layers to reduce spatial dimensions. Each has a filter size of 3x3 and a stride of 2.
- After the convolutional and max-pooling layers, there are three fully connected layers. The first two contain 4096 neurons and the last one contains a number of units equal to the number of classes (1000 in the original paper, as it was used on the ImageNet dataset), which, in our case, is 150.

- Dropout regularization is not explicitly included in the implementation in this project, whereas AlexNet applies dropout to the fully connected layers.
- The output layer employs the softmax activation function to generate class probabilities for the input image.

5. Custom CNN Implementation

The last model that was tried was a custom convolutional neural network, which follows this architecture:

- Four convolutional blocks are used with the following configuration:
  1. a 2-D convolution layer is applied with 64 filters, 3 by 3 kernel size and same padding and uses the ReLU activation function
  2. a batch normalization layer is then applied
  3. max pooling 2D is then applied, with a 2 by 2 pool size
- A fully connected layer of 512 neurons is applied to the flattened output of the last convolutional layer.
- An output layer with softmax activation is then applied to obtain the label predictions.

The model is compiled using the adam optimizer and categorical crossentropy as its loss function and uses the F1 score in addition to accuracy as training-time evaluation metrics.

6. Results

Surprisingly, the AlexNet and VGG16 implementations performed extremely poorly, with accuracies of 0.0398 and, respectively, 0.0099 on the test set. The custom cnn implementation, however, did much better as its accuracy was 0.5789 on the test set. Figures 2, 3 and 4 depict the confusion matrices for the three models on the test set.

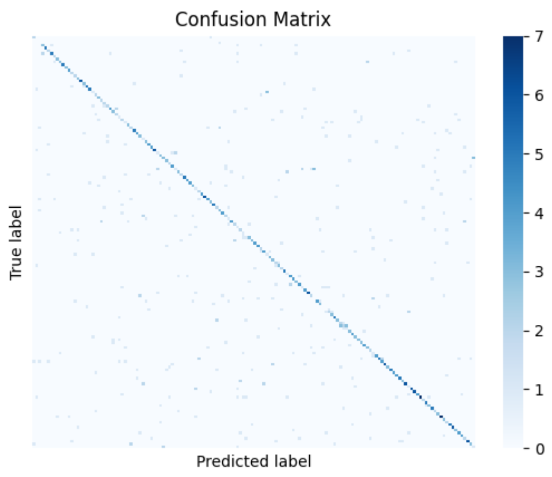

Figure 2. Custom CNN Confusion Matrix on the Test Set

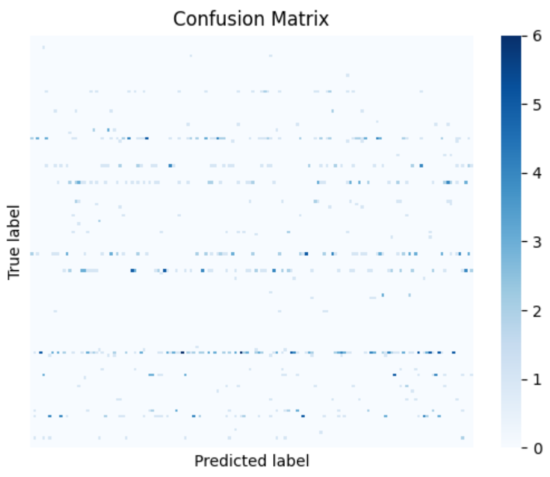

Figure 3. VGG16 Matrix on the Test Set

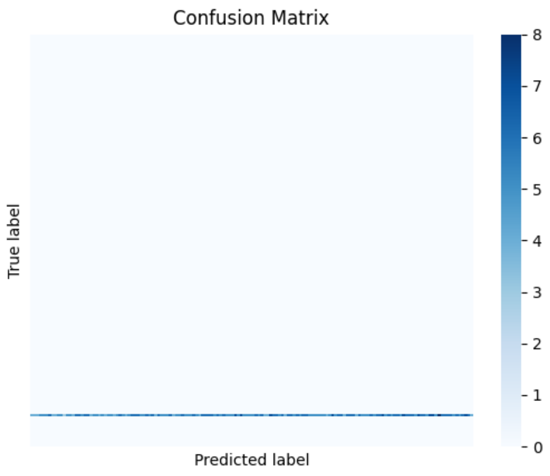

Figure 4. AlexNet Matrix on the Test Set

7. Conclusion

The technical analysis of the Custom CNN, VGG16 and AlexNet models reveals the fact that the former two were very unfit for this task, as the custom network gave much better predictions as opposed to the others.

References

[1] Simonyan, K. & Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. (2015)

[2] Krizhevsky, A., Sutskever, I. & Hinton, G. ImageNet classification with deep convolutional neural networks. *Proceedings Of The 25th International Conference On Neural Information Processing Systems - Volume 1.* pp. 1097-1105 (2012)