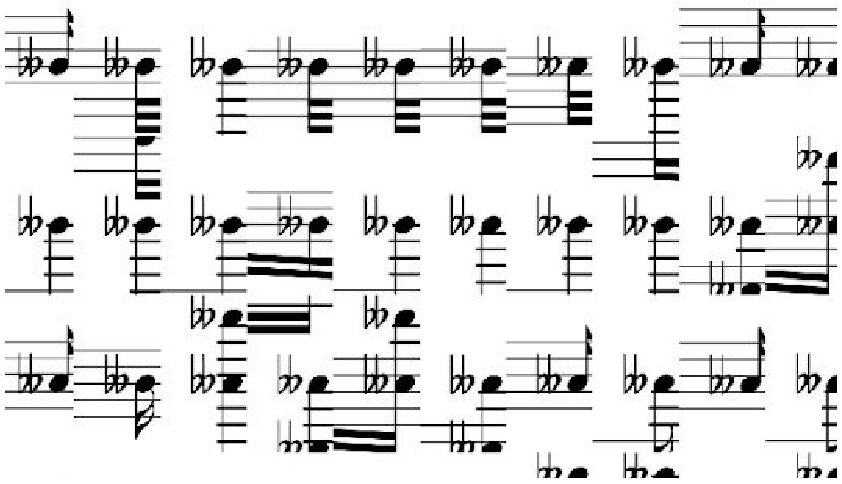
Music Object Classification: A statistical approach vs deep learning approach

Abi Pradhan

Music Object Classification is a part of Optical Music Recognition (OMR) where we task the computer to filter out and detect music symbols or objects in the music score sheet. In this project, I have simplified the problem down to classifying individual musical objects. I gathered a small subset of the DeepScores dataset, which had a collection of annotated labels for 118 musical objects. I will outline my approach to solving the problem using a simple statistical method and a deep learning method. In the process, I faced several challenges for handling large data but was able to get through most of them with work arounds involving standard data science practices. Finally, I will present my findings for both approaches.

Dataset:

For this problem, I used the DeepScores dataset. DeepScores is a collection of 300,000 sheets of music in digitized format. It comes along with annotation for music object classification, detection, and semantic segmentation. This makes it perfect for classification tasks using deep learning approaches. It is one of the few digitized collections of sheet music out there. For this problem, I used a subset of a dataset for object classification called DeepScores-Classification. DeepScores-classification is a collection of 124 classes of musical objects. Each class of music object is curated in separate folders containing digitized sheets. Each digitized sheet consists of processed individual music objects of that class that have been tiled together to make a sheet. The figure below should give an idea of digitized sheet format for an accidental double flat class .



Though there are 124 classes, 4 of them were empty and 2 had less than two individual music objects. So, there are 118 different objects that can be trained for classification. Processing all of these images gave me a dataset of 290,000 images of size 220 by 120 to play around with. Though this is not a very large image dataset, it is the perfect size for testing a simple Convolutional Neural Network without over capacitating the machine of facing extremely long wait times for training.

Challenges with the dataset:

The starter code to process the dataset, initially loaded all of the 290,000 individual objects into memory. While this may be possible on a powerful machine with large memory, it easily filled up the memory space on the machine I was testing on. A more standard approach is to process the images in batches. This would make it possible to train any CNN model of most machines. Infact, most deep learning frameworks have the option to train images on batches. I did this by storing file paths of all images along with its annotation and loading images in batches only for training the model. I used this idea for both the statistical and CNN for evaluation .

Statistical approach:

I used the Pearson correlation coefficient to measure similarities between the templates and test set. Just like we did in one of our labs, I calculated correlations between the templates and test images. First, I picked random templates for each class and second I picked averaged templates of 10 samples for each class. For both randomized and averaged templates, I first did a one-to-one template matching with an arbitrary threshold. Second, I found the best match template from all classes on each image in the test set. For both approaches, I got an interesting finding. The table below summarizes accuracies on several batch sizes. Here accuracy is simply defined as the percent of correct prediction to total samples.

Batch size	Randomized one-to-one	Averaged one-to-one	Randomised best match	Averaged best match
100	39	70.1	28.1	53.1
1000	30.1	72.7	25.2	61
10000	29.9	69.998	27	62

Running these experiments on bigger batch sizes is totally possible, but our findings above show that accuracies plateau on increasing batch sizes. In both the one-to-one and best match approach, accuracies bump up on averaging the templates.

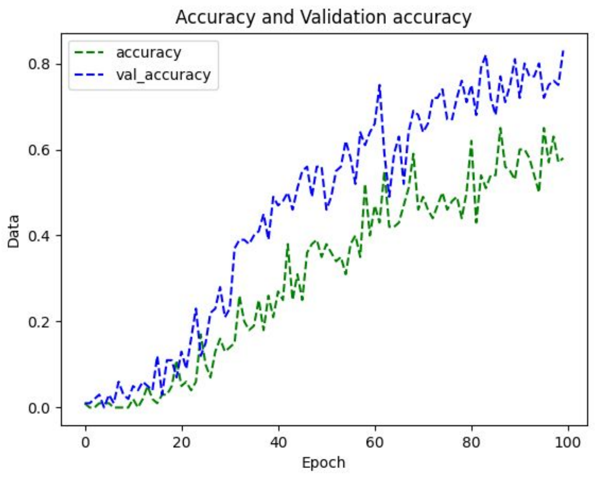
Deep Learning approach

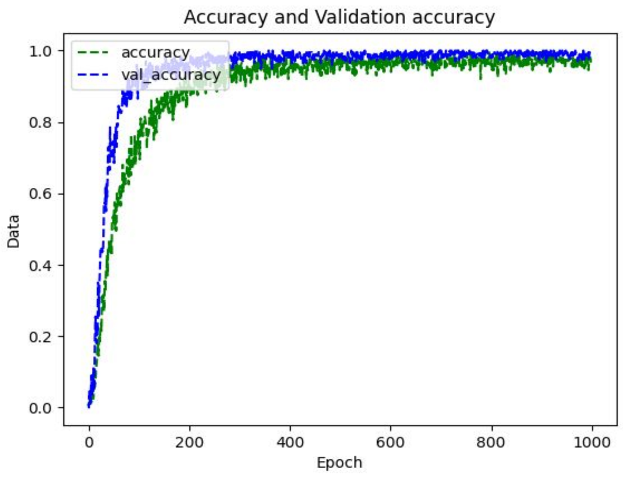
Next, I used a deep learning approach using Keras framework. I used a Convolutional Neural Network architecture that used to classify objects in the CIFAR-10, a dataset of 10 different classes like frog, car, aeroplane, bike. The architecture below summarizes the model:

```
        Layer (type)              Output Shape              Param #
=================================================================
    conv2d_1 (Conv2D)              (None, 220, 120, 32)      896
_________________________________________________________________
    activation_1 (Activation)      (None, 220, 120, 32)      0
_________________________________________________________________
    conv2d_2 (Conv2D)              (None, 218, 118, 32)      9248
_________________________________________________________________
    activation_2 (Activation)      (None, 218, 118, 32)      0
_________________________________________________________________
    max_pooling2d_1 (MaxPooling2 (None, 109, 59, 32)         0
_________________________________________________________________
    dropout_1 (Dropout)            (None, 109, 59, 32)       0
_________________________________________________________________
    conv2d_3 (Conv2D)              (None, 109, 59, 64)       18496
_________________________________________________________________
    activation_3 (Activation)      (None, 109, 59, 64)       0
_________________________________________________________________
    conv2d_4 (Conv2D)              (None, 107, 57, 64)       36928
_________________________________________________________________
    activation_4 (Activation)      (None, 107, 57, 64)       0
_________________________________________________________________
    max_pooling2d_2 (MaxPooling2 (None, 53, 28, 64)          0
_________________________________________________________________
    dropout_2 (Dropout)            (None, 53, 28, 64)        0
_________________________________________________________________
        flatten_1 (Flatten)        (None, 94976)             0
_________________________________________________________________
        dense_1 (Dense)            (None, 512)               48628224
_________________________________________________________________
    activation_5 (Activation)      (None, 512)               0
_________________________________________________________________
    dropout_3 (Dropout)            (None, 512)               0
_________________________________________________________________
        dense_2 (Dense)            (None, 118)               60534
=================================================================
                    Total params: 48,754,326
                  Trainable params: 48,754,326
                   Non-trainable params: 0
_________________________________________________________________
```

The dataset was first split in 7:2:1 ratio of train, validation and test sets. Minor adjustments like the input shape were made to fit images of size 220 by 120 by 3. The final output layer was filtered through to make predictions on the classes. Finally, the train set was put to training in batches to avoid memory overflow errors.

I ran several experiments with varying parameters like batch sizes, number of epochs and the number of steps. The model seems to do well without overfitting at a learning rate of 0.0001 and decay of





1e-6. The graphs below display accuracy and validation accuracy for two of my experiments. The one on the left is trained on batch size of 10 for 10 steps per epoch for 100 epochs, while the one on the right is trained for 1000 epochs. Both accuracy curves flattens out at 98 percent accuracy for the right experiment suggesting that there is no overfitting, Here val accuracy means accuracy on validation set and accuracy means accuracy on train set. Finally, running the trained model on the entire train set gave an accuracy of 92 percent suggesting the model is doing well.

Conclusion:

It was interesting to find a problem in computer vision and employ a deep learning method to solve the problem. In this problem, the CNN approach did much better. There are however flaws to this approach. First, the dataset is too processed and unrealistic because it is digitized and the consistency in the images in this dataset is simply not reflective of a real handwritten sheet music. That said, this does open doors for more complicated problems like segmenting objects in an actual sheet music. In fact, this is a problem with a good amount of research going onto it at the moment. DeepScores is one of the popular datasets with annotations available for it as well.