Deep learning Assignment

Name: Jessica Fornetti

Student Number:

Link to a zip file (not models):

https://drive.google.com/file/d/1EkbFpVZItgp56lKk8Q8DHTboGU6WeUvJ/view?usp=sharing

Link to the Models:

https://drive.google.com/file/d/1OXyroo2b8mbwWOu7DeqhXrMQZkGluGAk/view?usp=drive_link

https://drive.google.com/file/d/1JM4fj9Z5AL56WM6MrFxDs1wp9Coml98i/view?usp=drive_link

https://drive.google.com/file/d/1CX4n5D3QXihbEUtzx8EmEDdK8EKQ9pJ9/view?usp=drive_link

https://drive.google.com/file/d/1zBkwg0vOy_w7MiyVPpOd30RzmRu--h9d/view?usp=drive_link

“I confirm that the document and related code here is all my own work and that I did not engage in unfair practice in any way.”

PART 1 (Lyric Genre Prediction Task)

Data Preparation and Preprocessing:

The dataset of 'Multi-Lingual Lyrics for Genre Classification' from Kaggle is loaded. It has been down sampled as the dataset was taking too long to training the neural network models.

For the pre-processing step, the null values, stop words and punctuations were removed from our dataset. Then the dataset was split into 80% for training and 20% for validation.

Tokenization and Label Encoding on the target variables to convert the categorical features into integer values were performed.

A] (Models based only on Lyrics)

- RNN Variants 1) Basic RNN Model:

```
Epoch 9/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 38s 434ms/step - accuracy: 0.7727 - loss: 0.8376 - val_accuracy: 0.5050 - val_loss: 1.9194
Epoch 10/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 21s 441ms/step - accuracy: 0.7862 - loss: 0.7801 - val_accuracy: 0.5048 - val_loss: 1.9687
Epoch 11/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 41s 438ms/step - accuracy: 0.7995 - loss: 0.7119 - val_accuracy: 0.5000 - val_loss: 2.0247
```

2) Single Layer LSTM Model:

```
Epoch 4/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 41s 879ms/step - accuracy: 0.7862 - loss: 0.6484 - val_accuracy: 0.4950 - val_loss: 1.4512
Epoch 5/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 42s 886ms/step - accuracy: 0.8274 - loss: 0.5365 - val_accuracy: 0.4237 - val_loss: 1.5180
Epoch 6/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 79s 828ms/step - accuracy: 0.8661 - loss: 0.4376 - val_accuracy: 0.5265 - val_loss: 1.5184
```

3) Multi-Layer LSTM Model:

```
Epoch 9/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 73s 2s/step - accuracy: 0.7266 - loss: 0.9355 - val_accuracy: 0.5480 - val_loss: 1.3976
Epoch 10/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 73s 2s/step - accuracy: 0.7412 - loss: 0.8878 - val_accuracy: 0.5533 - val_loss: 1.4110
Epoch 11/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 71s 2s/step - accuracy: 0.7600 - loss: 0.8340 - val_accuracy: 0.5595 - val_loss: 1.4275
```

Out of the three variants of the RNN model we can see that Single-Layer LSTM model gives around 86% of training accuracy but only 52% validation accuracy, meaning the model is likely overfitting. Similarly, the basic RNN model obtains around 79% of training accuracy and 50% of validation accuracy.

The Multi-Layer LSTM model performs better as compared to the two other models as it gives around 76% accuracy and 56% validation accuracy, which indicates it is overfitting. To increase the model's accuracy, we could increase the model complexity by adding more LSTM layers. In addition, we could try using different dropout rates for reducing overfitting even more.

Moreover, the accuracy obtained are all quite similar as the same state_size was used for all 3 models.

- Embeddings
    1) On the fly embeddings:

```
Epoch 7/20
1451/1451 ━━━━━━━━━━━━━━━━━━━━ 41s 28ms/step - accuracy: 0.6989 - loss: 0.8789 - val_accuracy: 0.6143 - val_loss: 1.1555
Epoch 8/20
1451/1451 ━━━━━━━━━━━━━━━━━━━━ 42s 28ms/step - accuracy: 0.7088 - loss: 0.8379 - val_accuracy: 0.6132 - val_loss: 1.1758
Epoch 9/20
1451/1451 ━━━━━━━━━━━━━━━━━━━━ 44s 30ms/step - accuracy: 0.7260 - loss: 0.7919 - val_accuracy: 0.6146 - val_loss: 1.2165
```

    2) Pre trained embeddings:

```
Epoch 10/20
1451/1451 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.5699 - loss: 1.1884 - val_accuracy: 0.5655 - val_loss: 1.2162
Epoch 11/20
1451/1451 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.5718 - loss: 1.1860 - val_accuracy: 0.5640 - val_loss: 1.2124
Epoch 12/20
1451/1451 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.5731 - loss: 1.1757 - val_accuracy: 0.5656 - val_loss: 1.2131
```

We can see that on-the-fly embeddings have overall higher accuracy on training and validation (72% and 61%), however pre-trained embeddings have less accuracy as compared to the other model but are less prone to overfitting.

The on-the-fly embeddings might capture the variations of the lyrics in the dataset better the pre-trained embeddings which are trained on larger corpus and therefore are more general. Another major difference is the dimensionality of the embeddings given that the on-the-fly embeddings have 100 dimensions while the pre-trained ones only have 50 dimensions.

    3) Traditional text encoding approach:

TF-IDF Accuracy: 0.6068412889884542

Similarly, as the on-the-fly embeddings, TF-IDF is able to capture more important and relevant features as it is trained directly on the data which allows the logistic regression to achieve a better performance.

- CNN for Text Classification

    1) CNNs with same kernel sizes:

```
Epoch 15/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 39s 837ms/step - accuracy: 0.8978 - loss: 0.3141 - val_accuracy: 0.5286 - val_loss: 1.9337
Epoch 16/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 43s 872ms/step - accuracy: 0.9052 - loss: 0.2938 - val_accuracy: 0.5203 - val_loss: 2.0809
Epoch 17/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 39s 815ms/step - accuracy: 0.9142 - loss: 0.2654 - val_accuracy: 0.5615 - val_loss: 2.1487
```

    2) CNNs with different kernel sizes:

```
Epoch 5/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 82s 886ms/step - accuracy: 0.6616 - loss: 1.0048 - val_accuracy: 0.0698 - val_loss: 2.5508
Epoch 6/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 41s 878ms/step - accuracy: 0.7087 - loss: 0.8827 - val_accuracy: 0.0699 - val_loss: 2.8057
Epoch 7/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 42s 892ms/step - accuracy: 0.7366 - loss: 0.7819 - val_accuracy: 0.0714 - val_loss: 2.9371
```

We can see that the CNN model with same kernel size obtains a better performance as it captures more local features consistently on text data and it also simplifies the architecture of our model making it more generalizable than compared to the CNN model with different kernel sizes.

3) CNN as an additional layer before a LSTM solution:

1st method:

```
Epoch 11/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 43s 911ms/step - accuracy: 0.7780 - loss: 0.8124 - val_accuracy: 0.5663 - val_loss: 1.6999
Epoch 12/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 44s 928ms/step - accuracy: 0.7997 - loss: 0.7480 - val_accuracy: 0.5859 - val_loss: 1.6330
Epoch 13/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 79s 865ms/step - accuracy: 0.8077 - loss: 0.7130 - val_accuracy: 0.5624 - val_loss: 1.8929
```

2nd method:

```
Epoch 8/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 167s 4s/step - accuracy: 0.6782 - loss: 1.1578 - val_accuracy: 0.3718 - val_loss: 2.7453
Epoch 9/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 162s 3s/step - accuracy: 0.6972 - loss: 1.0949 - val_accuracy: 0.3771 - val_loss: 3.2613
Epoch 10/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 159s 3s/step - accuracy: 0.7084 - loss: 1.0566 - val_accuracy: 0.4073 - val_loss: 2.4415
```

Here we can see that the 1st method gives better results (80% accuracy, 56% val accuracy) as compared to the second method (70% accuracy ,40% val accuracy) given that we don't flatten the output after 3rd convolutional layer before giving it to the LSTM layer for 1st method, so we don't lose any information.

Compared to all the other neural network models we can see that CNN overfits a lot.

- Comparison to Non-Neural Methods
  A Decision Tree model was used. However, as the model architecture of Decision Tree is not that complex it does not capture relevant enough features of the textual data even after tuning the hyper parameters, as seen below.

  Decision Tree : Accuracy: 0.3246596587971739

B] (Models based on Lyrics and Artist)

The artist feature was added to provide additional information, and all models were rebuilt and retrained with that additional feature.

- RNN Variants Basic RNN Model:

```
Epoch 12/20
47/47 [==============================] - 20s 424ms/step - loss: 1.3700 - accuracy: 0.4459 - val_loss: 1.4938 - val_accuracy: 0.
3704
Epoch 13/20
47/47 [==============================] - 20s 430ms/step - loss: 1.3617 - accuracy: 0.4474 - val_loss: 1.5511 - val_accuracy: 0.
4166
Epoch 14/20
47/47 [==============================] - 22s 464ms/step - loss: 1.3629 - accuracy: 0.4472 - val_loss: 1.5625 - val_accuracy: 0.
4163
```

Single Layer LSTM Model:

```
Epoch 5/20
47/47 [==============================] - 47s 1s/step - loss: 0.8197 - accuracy: 0.7258 - val_loss: 1.4819 - val_accuracy: 0.411
8
Epoch 6/20
47/47 [==============================] - 41s 883ms/step - loss: 0.7103 - accuracy: 0.7670 - val_loss: 1.5365 - val_accuracy: 0.
4138
Epoch 7/20
47/47 [==============================] - 42s 891ms/step - loss: 0.6300 - accuracy: 0.7960 - val_loss: 1.6264 - val_accuracy: 0.
4125
```

Multi-Layer LSTM Model:

```
Epoch 12/20
47/47 [==============================] - 66s 1s/step - loss: 1.3732 - accuracy: 0.4492 - val_loss: 1.4600 - val_accuracy: 0.4184
Epoch 13/20
47/47 [==============================] - 64s 1s/step - loss: 1.3702 - accuracy: 0.4490 - val_loss: 1.4604 - val_accuracy: 0.4192
Epoch 14/20
47/47 [==============================] - 61s 1s/step - loss: 1.3673 - accuracy: 0.4488 - val_loss: 1.4684 - val_accuracy: 0.4196
```

Adding the Artist feature adds noise to the models as certain artist might not belong to a single genre. The Single Layer LSTM model overfits a lot as compared to other two models (with a 63% training accuracy and 41% validation accuracy). It seems like the Basic RNN Model and Multi-Layer LSTM accuracies could be improved as they don't seem too overfit much.

- Embeddings

1) On the fly embeddings:

```
Epoch 7/20
1451/1451 ━━━━━━━━━━━━━━━━━━━━ 65s 24ms/step - accuracy: 0.4627 - loss: 1.7408 - val_accuracy: 0.3948 - val_loss: 1.9427
Epoch 8/20
1451/1451 ━━━━━━━━━━━━━━━━━━━━ 53s 32ms/step - accuracy: 0.4801 - loss: 1.6683 - val_accuracy: 0.4092 - val_loss: 2.1516
Epoch 9/20
1451/1451 ━━━━━━━━━━━━━━━━━━━━ 89s 37ms/step - accuracy: 0.5038 - loss: 1.5493 - val_accuracy: 0.3583 - val_loss: 2.0047
```

2) Pretrained embeddings:

```
Epoch 18/20
1451/1451 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.7901 - loss: 0.6796 - val_accuracy: 0.7656 - val_loss: 0.7676
Epoch 19/20
1451/1451 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.7920 - loss: 0.6753 - val_accuracy: 0.7693 - val_loss: 0.7593
Epoch 20/20
1451/1451 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.7960 - loss: 0.6655 - val_accuracy: 0.7766 - val_loss: 0.7402
```

We can see here that the pre-trained embedding model has performed better than the on-fly embeddings model. This may be due to the difference in how we encode the artist feature, as for the on-the-fly embeddings we encode the artist feature as an int whereas for the pre trained embeddings model we also generate an embedding for the artist.

3) Traditional text approach:

TF-IDF Accuracy: 0.6313113906600034

For TF-IDF we obtain a similar accuracy when trained on the lyrics and the artist. Adding the artist feature made a slight improvement in the accuracy of +3%.

- CNN for Text Classification

1) CNNs with same kernel sizes:

```
Epoch 7/20
47/47 [==============================] - 41s 877ms/step - loss: 1.2582 - accuracy: 0.5155 - val_loss: 1.6171 - val_accuracy: 0.
3702
Epoch 8/20
47/47 [==============================] - 41s 869ms/step - loss: 1.1771 - accuracy: 0.5540 - val_loss: 1.6401 - val_accuracy: 0.
3761
Epoch 9/20
47/47 [==============================] - 41s 880ms/step - loss: 1.0937 - accuracy: 0.5907 - val_loss: 1.8266 - val_accuracy: 0.
3770
```

2) CNNs with different kernel sizes:

```
Epoch 9/20
47/47 [==============================] - 45s 952ms/step - loss: 1.1567 - accuracy: 0.5004 - val_loss: 1.5872 - val_accuracy: 0.
3945
Epoch 10/20
47/47 [==============================] - 41s 880ms/step - loss: 1.0815 - accuracy: 0.5491 - val_loss: 1.7017 - val_accuracy: 0.
3261
Epoch 11/20
47/47 [==============================] - 43s 920ms/step - loss: 0.9976 - accuracy: 0.5957 - val_loss: 2.0739 - val_accuracy: 0.
1972
```

As stated earlier, the CNN models with different kernel sizes gives poor results compared to the same kernel size models due the model architecture it uses.

CNN as an additional layer before a LSTM solution:

1st method:

```
Epoch 12/20
47/47 [==============================] - 42s 905ms/step - loss: 1.3074 - accuracy: 0.5623 - val_loss: 1.6211 - val_accuracy:
0.3740
Epoch 13/20
47/47 [==============================] - 42s 890ms/step - loss: 1.2641 - accuracy: 0.5910 - val_loss: 1.6054 - val_accuracy:
0.3907
Epoch 14/20
47/47 [==============================] - 42s 895ms/step - loss: 1.2192 - accuracy: 0.6192 - val_loss: 1.6520 - val_accuracy:
0.3970
```

2nd method:

```
Epoch 9/20
47/47 [==============================] - 201s 4s/step - loss: 1.3515 - accuracy: 0.4607 - val_loss: 1.7897 - val_accuracy: 0.41
06
Epoch 10/20
47/47 [==============================] - 195s 4s/step - loss: 1.3383 - accuracy: 0.4589 - val_loss: 1.9315 - val_accuracy: 0.40
97
Epoch 11/20
47/47 [==============================] - 219s 5s/step - loss: 1.3336 - accuracy: 0.4630 - val_loss: 1.9929 - val_accuracy: 0.41
99
```

We can notice that the 2nd method is overfitting less and slightly better, however compared to the lyrics only (section A), the accuracy of this section is worse.

- Comparison to Non-Neural Methods

  Here we can see that the Decision Tree continues to perform terribly and worse than the Neural Network methods.

  Decision Tree: Accuracy: 0.29691538859210753

To conclude, the results of section A (with lyrics only) were better in terms of accuracy and overfitting than the results of section B (with lyrics and artist). The only difference in result was observed for the embeddings section, where the pre-trained embedding performed better for section B than section A most likely due to the difference in the way of encoding the artist feature.

PART 2 (Image Processing and Transfer Learning)

Data Preparation and Preprocessing:

The CIFAR-100 dataset from the keras dataset module was imported.
The dataset was randomly split into two blocks (Block 1 and Block 2) which include 50 classes each.
x_block1 → contains image data in the form of a 2D array containing 50 randomly selected classes.
y_block1 → contains the target variable of the 50 selected classes.
x_block2 → contains image data in the form of a 2D array containing the remaining 50 classes.
y_block2 → contains the target variable of the 50 remaining classes

Basic Modelling: For Block 1 Images

Multiple versions of CNN models were built by using different activation function, skip connection and different loss function.

1) Basic CNN Model:
   We can see that our model is giving 64% accuracy, however there is a slight difference between the training and the validation accuracy. The model does not seem to be overfitting a lot. Using different dropout rates and increasing the number of filters and the nodes of the network could lead to an increase in accuracy.

```
Epoch 20/40
625/625 [==============================] - 118s 189ms/step - loss: 1.1806 - accuracy: 0.6380 - val_loss: 1.8743 - val_accuracy: 0.5110 - lr: 4.0000e-05
Epoch 21/40
625/625 [==============================] - 112s 179ms/step - loss: 1.1664 - accuracy: 0.6401 - val_loss: 1.8879 - val_accuracy: 0.5106 - lr: 4.0000e-05
Epoch 22/40
625/625 [==============================] - 111s 178ms/step - loss: 1.1445 - accuracy: 0.6498 - val_loss: 1.8791 - val_accuracy: 0.5130 - lr: 8.0000e-06
Epoch 23/40
625/625 [==============================] - 113s 181ms/step - loss: 1.1432 - accuracy: 0.6480 - val_loss: 1.8806 - val_accuracy: 0.5138 - lr: 8.0000e-06
```

2) Adding tanh activation function in the hidden layers:
   We can see that the model is not overfitting however, the accuracy is very poor when using the tanh function in the hidden layers. Moreover, the model reached around 40% of accuracy with little variation in 40 epochs. As we know, the tanh function can lead to problems like vanishing gradient and internal covariate shift, which seems to be the case given that the accuracy seems to be constant over the 40 epochs.

```
Epoch 37/40
625/625 [==============================] - 113s 181ms/step - loss: 2.2212 - accuracy: 0.3929 - val_loss: 2.3819 - val_accuracy: 0.3584 - lr: 4.0000e-05
Epoch 38/40
625/625 [==============================] - 113s 180ms/step - loss: 2.2174 - accuracy: 0.3956 - val_loss: 2.3872 - val_accuracy: 0.3560 - lr: 4.0000e-05
Epoch 39/40
625/625 [==============================] - 117s 187ms/step - loss: 2.2076 - accuracy: 0.3971 - val_loss: 2.3783 - val_accuracy: 0.3596 - lr: 8.0000e-06
Epoch 40/40
625/625 [==============================] - 112s 179ms/step - loss: 2.2066 - accuracy: 0.3952 - val_loss: 2.3801 - val_accuracy: 0.3572 - lr: 8.0000e-06
<keras.src.callbacks.History at 0x7dbbcca9e800>
```

3) Adding the Logistic/Sigmoid activation function in the final layer:
   We can see that the accuracy reaches 72%, however the model is overfitting as highlighted by the validation accuracy. As we know, that sigmoid function squishes the output between 0 and 1 which makes it suitable for binary classification rather than predicting the multi-class output. This most likely the reason why our model is overfitting too much given that the activation function is not adapted.

```
Epoch 22/40
625/625 [==============================] - 118s 189ms/step - loss: 0.9612 - accuracy: 0.7088 - val_loss: 1.9111 - val_accuracy: 0.5222 - lr: 2.0000e-04
Epoch 23/40
625/625 [==============================] - 117s 187ms/step - loss: 0.9002 - accuracy: 0.7189 - val_loss: 1.9089 - val_accuracy: 0.5220 - lr: 4.0000e-05
Epoch 24/40
625/625 [==============================] - 116s 186ms/step - loss: 0.8864 - accuracy: 0.7201 - val_loss: 1.9168 - val_accuracy: 0.5198 - lr: 4.0000e-05
<keras.src.callbacks.History at 0x7dbbcc7cbc70>
```

4) Adding skip connection in the Model:
   We can notice that among all the CNN models, the skip connection model gave the highest accuracy, although it is overfitting a bit. Given that the model contains only 5 layers and skip connection performs best with the deeper neural network models, this may be one of the reasons that explain the overfitting. Therefore, increasing the complexity of the model using the skip connection could reduce overfitting drastically.

```
Epoch 16/40
625/625 [==============================] - 195s 313ms/step - loss: 1.0528 - accuracy: 0.6528 - val_loss: 2.0483 - val_accuracy: 0.4946 - lr: 2.0000e-04
Epoch 17/40
625/625 [==============================] - 195s 313ms/step - loss: 0.9882 - accuracy: 0.6743 - val_loss: 2.0995 - val_accuracy: 0.5018 - lr: 2.0000e-04
Epoch 18/40
625/625 [==============================] - 207s 331ms/step - loss: 0.9048 - accuracy: 0.6971 - val_loss: 2.2044 - val_accuracy: 0.4950 - lr: 2.0000e-04
Epoch 19/40
625/625 [==============================] - 193s 309ms/step - loss: 0.8256 - accuracy: 0.7172 - val_loss: 2.1501 - val_accuracy: 0.5056 - lr: 4.0000e-05
Epoch 20/40
625/625 [==============================] - 193s 308ms/step - loss: 0.8053 - accuracy: 0.7256 - val_loss: 2.1863 - val_accuracy: 0.5052 - lr: 4.0000e-05
```

Autoencoder Modelling: For Block 1 Images

As we can see that the accuracy is poor, however the loss function is decreasing. The training accuracy is around 46-51% while the validation accuracy is slightly lower, but close to the training accuracy indicating that the model is not overfitting as much. The accuracy could be increased by increasing the model complexity and by training on more epochs.

The mean_squared_error loss function along with the sigmoid activation function were used. Filters in the convolutional layer (32,64) along with 128 and 64 neurons followed in the dense layer were used to increase the accuracy. Regularization was used to prevent the overfitting and Batch Normalization was used to reduce the problem of Vanishing gradient. In addition, Up sampling was used to improve the feature representation.

```
Epoch 11/15
157/157 [==============================] - 122s 778ms/step - loss: 20045.2871 - accuracy: 0.4498 - val_loss: 20319.0938 - val_accuracy: 0.4630
Epoch 12/15
157/157 [==============================] - 124s 787ms/step - loss: 20045.2832 - accuracy: 0.4554 - val_loss: 20319.0938 - val_accuracy: 0.4651
Epoch 13/15
157/157 [==============================] - 123s 786ms/step - loss: 20045.3066 - accuracy: 0.4565 - val_loss: 20319.0957 - val_accuracy: 0.4236
Epoch 14/15
157/157 [==============================] - 120s 762ms/step - loss: 20045.2793 - accuracy: 0.4643 - val_loss: 20319.0938 - val_accuracy: 0.4560
Epoch 15/15
157/157 [==============================] - 119s 756ms/step - loss: 20045.2754 - accuracy: 0.4671 - val_loss: 20319.0898 - val_accuracy: 0.4685
<keras.src.callbacks.History at 0x790d61a84eb0>
```

Transfer Learning: For Block 2 Images

The CNN and the autoencoder pre-trained model were combined to build a transfer learning model. The training accuracy of the model is around 87%, however the model seems to be overfitting given the drastic the difference with the validation accuracy. Additionally, the model gave 87% accuracy in just 16 epochs which shows the improved potential performance of the model.
The dropout rate of 0.5 was used for the fully connected layers. A Dense layer of 256 neurons was used. As compared to the results of the single CNN and autoencoder model, the transfer learning model performs much better as shown by the improved feature extraction and reduced training time.

```
Epoch 13/50
625/625 [==============================] - 247s 396ms/step - loss: 0.9920 - accuracy: 0.7187 - val_loss: 2.3290 - val_accuracy: 0.4728 - lr: 0.0010
Epoch 14/50
625/625 [==============================] - 262s 420ms/step - loss: 0.8785 - accuracy: 0.7487 - val_loss: 2.2702 - val_accuracy: 0.4878 - lr: 0.0010
Epoch 15/50
625/625 [==============================] - 239s 382ms/step - loss: 0.5626 - accuracy: 0.8387 - val_loss: 2.1548 - val_accuracy: 0.5494 - lr: 2.0000e-04
Epoch 16/50
625/625 [==============================] - 241s 385ms/step - loss: 0.4207 - accuracy: 0.8781 - val_loss: 2.3072 - val_accuracy: 0.5490 - lr: 2.0000e-04
<keras.src.callbacks.History at 0x796a28867e50>
```

Comparison of CNN Model and Autoencoder Model on Block2 Images

1) CNN Model on Block2 Images -
   As compared to the transfer learning model this model gives a lower accuracy on the block 2 images dataset. Nonetheless, the model gives a good accuracy, although it is overfitting a bit.

```
Epoch 17/40
625/625 [==============================] - 228s 365ms/step - loss: 0.9930 - accuracy: 0.6744 - val_loss: 1.9332 - val_accuracy: 0.5352 - lr: 2.0000e-04
Epoch 18/40
625/625 [==============================] - 236s 377ms/step - loss: 0.9150 - accuracy: 0.7037 - val_loss: 1.9457 - val_accuracy: 0.5400 - lr: 2.0000e-04
Epoch 19/40
625/625 [==============================] - 206s 330ms/step - loss: 0.8463 - accuracy: 0.7147 - val_loss: 1.9717 - val_accuracy: 0.5448 - lr: 4.0000e-05
Epoch 20/40
625/625 [==============================] - 214s 342ms/step - loss: 0.8129 - accuracy: 0.7304 - val_loss: 2.0028 - val_accuracy: 0.5390 - lr: 4.0000e-05
<keras.src.callbacks.History at 0x796a1ef7cbe0>
```

2) Autoencoder Model on Block2 Images -
   We can see that the model is not overfitting, however to increase accuracy the model should be trained on more epochs the model complexity should be increased.

```
Epoch 11/15
125/125 [==============================] - 113s 902ms/step - loss: 19930.1133 - accuracy: 0.5209 - val_loss: 19944.2324 - val_accuracy: 0.5353
Epoch 12/15
125/125 [==============================] - 117s 934ms/step - loss: 19930.1035 - accuracy: 0.5133 - val_loss: 19944.2227 - val_accuracy: 0.5013
Epoch 13/15
125/125 [==============================] - 116s 930ms/step - loss: 19930.1055 - accuracy: 0.5086 - val_loss: 19944.2207 - val_accuracy: 0.5119
Epoch 14/15
125/125 [==============================] - 116s 931ms/step - loss: 19930.1094 - accuracy: 0.5046 - val_loss: 19944.2207 - val_accuracy: 0.5019
Epoch 15/15
125/125 [==============================] - 118s 942ms/step - loss: 19930.1035 - accuracy: 0.5008 - val_loss: 19944.2207 - val_accuracy: 0.5028
```

To conclude, the transfer learning model performs well leading to improved performance, faster convergence and better generalization than two pretrained models as it allows the transfer of knowledge and representations learned from one task to another and can capture better features.