Malware Analysis Project

Comsats University Islamabad

FA21-BCT-021	**Waseeq Ur Rehman**

Project Overview:

The project aims to use API columns as sequences and apply Long Short-Term Memory (LSTM) neural networks for classification. The dataset used in this project can be found at [CIC-ANDMal2020](). The goal is to classify sequences of API calls into different malware categories.

Code Explanation:

1. **Imports**: The necessary libraries are imported, including pandas for data manipulation, glob for file operations, scikit-learn for preprocessing and model evaluation, TensorFlow for building and training neural networks, and Matplotlib for visualization.
2. **Load Data Function**: The `load_data` function reads data from a CSV file and adds label columns to the DataFrame. Labels are extracted from the file names.
3. **Model Creation Function**: The `create_model` function defines the architecture of the LSTM model. It consists of two LSTM layers followed by dropout layers to prevent overfitting. The output layer is a dense layer with softmax activation for multi-class classification.
4. **Read and Combine Data**: The code reads multiple CSV files containing sequential data. It extracts label information from the file names and combines the data into a single DataFrame.
5. **Preprocessing**: The input data is scaled using Min-Max scaling to normalize it. Sequences are padded to ensure uniform length, necessary for feeding them into the LSTM model.
6. **Split Data**: The data is split into training and testing sets using train_test_split from scikit-learn.
7. **Model Training**: The LSTM model is created using the `create_model` function. It is compiled with the Adam optimizer and categorical cross-entropy loss. The model is trained on the training data, and the training history is stored for visualization.
8. **Evaluation**: The trained model is evaluated on the test data to assess its performance in terms of loss and accuracy.
9. **Visualization**: Matplotlib is used to visualize training and validation loss and accuracy, as well as the average accuracy and loss across folds in K-Fold cross-validation.

Model Details

- **LSTM Layers**: Two LSTM layers are used with 128 and 64 units, respectively. The first layer processes input sequences and returns sequences, while the second layer processes the sequences outputted by the first layer and returns a single output for each sequence.
- **Dropout Layers**: Two dropout layers with a dropout rate of 0.5 are added after each LSTM layer to prevent overfitting.
- **Output Layer**: The output layer is a dense (fully connected) layer with softmax activation, producing the final output of the model, which is a probability distribution over the different classes.

Working of the Code

1. Load data from CSV files, extracting labels.
2. Preprocess the data by scaling and padding sequences.
3. Split the data into training and testing sets.
4. Create and train the LSTM model.
5. Evaluate the model on the test data to assess its performance.
6. Visualize training and validation metrics.
7. Perform K-Fold cross-validation to further validate the model's performance.
8. Visualize the average accuracy and loss across folds.

This comprehensive approach allows for the development of a robust LSTM model for classifying sequences of API calls into different malware categories.

Code

```
import pandas as pd
import glob
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from sklearn.model_selection import KFold, train_test_split
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam
import numpy as np
import matplotlib.pyplot as plt

def load_data(file_path, label):
    df = pd.read_csv(file_path)
    root_label = label.split('_')[0]
    # Assign the root label to all instances in the DataFrame
    df['label'] = root_label
    return df


def create_model(input_shape, num_classes):
    model = Sequential()
    model.add(LSTM(128, input_shape=input_shape,
return_sequences=True))
    model.add(Dropout(0.5))
    model.add(LSTM(64))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))  # Use softmax
for multi-class classification

    model.compile(optimizer=Adam(learning_rate=0.001),
loss='categorical_crossentropy', metrics=['accuracy'])
    return model


# Assuming this is the path to your sample CSV file
sample_csv_path = 'Adware_after_reboot_Cat.csv'

# Read the sample CSV to get the column names
sample_df = pd.read_csv(sample_csv_path)
api_columns = [col for col in sample_df.columns if
col.startswith('API_')]
print(api_columns)


combined_df = pd.DataFrame()
file_paths = glob.glob('/content/*.csv')
```

```
num_files = len(file_paths)
print("Number of files found:", num_files)

for file_path in file_paths:
    file_name = file_path.split('/')[-1]  # Extract file name from path
    try:
        label, reboot_status = file_name.split('_Cat')[0].rsplit('_',
1)
    except ValueError:
        print("Error processing file:", file_name)
        continue
    label = label
    df = load_data(file_path, label)
    combined_df = pd.concat([combined_df, df], ignore_index=True)


X = combined_df[api_columns]
y = combined_df['label']
class_counts = y.value_counts()

# Print the number of samples for each class
print("Number of samples for each class:")
print(class_counts)


label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

X_seq = pad_sequences(X_scaled, maxlen=len(api_columns),
dtype='float32', padding='post', truncating='post')

X_seq = X_seq.reshape((X_seq.shape[0], 1, X_seq.shape[1]))


num_classes = len(combined_df['label'].unique())

# Convert labels to one-hot encoded format
y_encoded = to_categorical(y_encoded, num_classes=num_classes)

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_seq, y_encoded,
test_size=0.2, random_state=42)

# Create and train the model
model = create_model((1, len(api_columns)), num_classes)
```

```
history = model.fit(X_train, y_train, epochs=50, batch_size=128,
validation_split=0.2)

# Evaluate on test data
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test Loss: {loss:.4f}, Test Accuracy: {accuracy:.4f}')
train_loss = history.history['loss']
train_accuracy = history.history['accuracy']
val_loss = history.history['val_loss']
val_accuracy = history.history['val_accuracy']
epochs = range(1, len(train_loss) + 1)

# Plotting
plt.figure(figsize=(12, 5))

# Plotting loss
plt.subplot(1, 2, 1)
plt.plot(epochs, train_loss, 'b', label='Training Loss')
plt.plot(epochs, val_loss, 'r', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

# Plotting accuracy
plt.subplot(1, 2, 2)
plt.plot(epochs, train_accuracy, 'b', label='Training Accuracy')
plt.plot(epochs, val_accuracy, 'r', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

Output

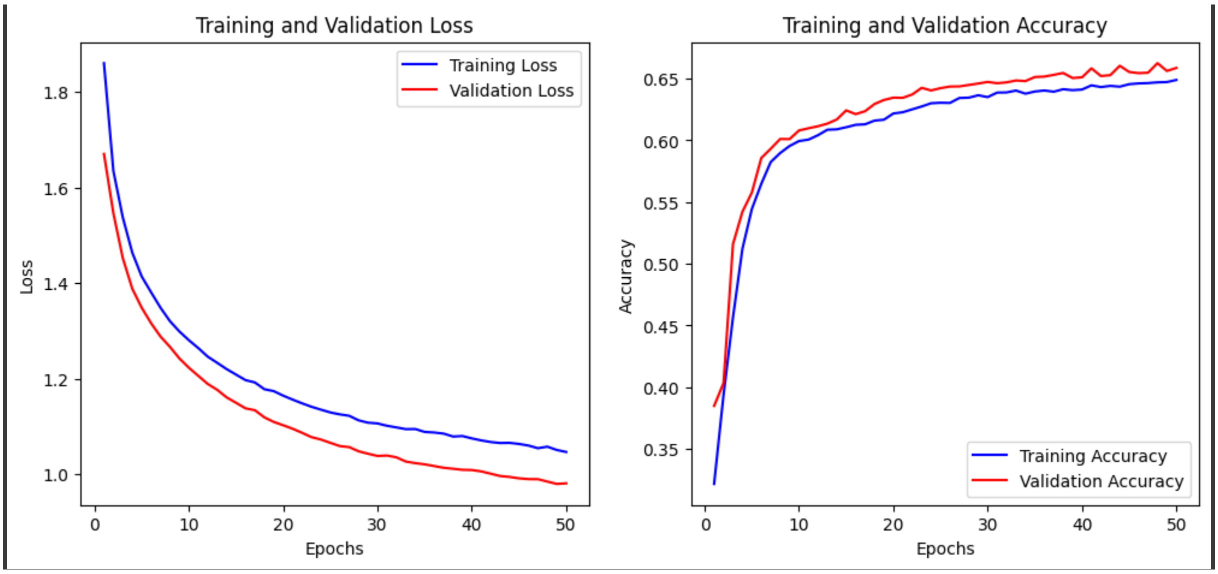
```
['API_Process_android.os.Process_start',
'API_Process_android.app.ActivityManager_killBackgroundProcesses',
'API_Process_android.os.Process_killProcess',
'API_Command_java.lang.Runtime_exec', . . . . . .
Number of files found: 28
Number of samples for each class:
label
Trojan          15027
Riskware        14053
Adware          10980
Zero             4475
Ransomware       3411
NoCategory       1932
```

```
PUA               1290
Backdoor          1137
Scareware          886
FileInfector       248
Name: count, dtype: int64
Epoch 1/50
268/268 [==============================] - 15s 15ms/step - loss: 1.8605 -
accuracy: 0.3217 - val_loss: 1.6705 - val_accuracy: 0.3849
Epoch 2/50
268/268 [==============================] - 3s 11ms/step - loss: 1.6346 -
accuracy: 0.3939 - val_loss: 1.5459 - val_accuracy: 0.4035
Epoch 3/50
268/268 [==============================] - 3s 13ms/step - loss: 1.5372 -
accuracy: 0.4566 - val_loss: 1.4520 - val_accuracy: 0.5160
Epoch 4/50
268/268 [==============================] - 5s 19ms/step - loss: 1.4638 -
accuracy: 0.5121 - val_loss: 1.3883 - val_accuracy: 0.5422
Epoch 5/50
268/268 [==============================] - 3s 11ms/step - loss: 1.4139 -
accuracy: 0.5444 - val_loss: 1.3483 - val_accuracy: 0.5577
Epoch 6/50
268/268 [==============================] - 3s 11ms/step - loss: 1.3806 -
accuracy: 0.5647 - val_loss: 1.3158 - val_accuracy: 0.5857
Epoch 7/50
268/268 [==============================] - 3s 12ms/step - loss: 1.3482 -
accuracy: 0.5824 - val_loss: 1.2878 - val_accuracy: 0.5931
Epoch 8/50
268/268 [==============================] - 5s 20ms/step - loss: 1.3201 -
accuracy: 0.5897 - val_loss: 1.2663 - val_accuracy: 0.6011
Epoch 9/50
268/268 [==============================] - 3s 12ms/step - loss: 1.2986 -
accuracy: 0.5954 - val_loss: 1.2421 - val_accuracy: 0.6010
Epoch 10/50
268/268 [==============================] - 3s 12ms/step - loss: 1.2804 -
accuracy: 0.5994 - val_loss: 1.2227 - val_accuracy: 0.6080
Epoch 11/50
268/268 [==============================] - 3s 12ms/step - loss: 1.2639 -
accuracy: 0.6006 - val_loss: 1.2057 - val_accuracy: 0.6098
Epoch 12/50
268/268 [==============================] - 5s 18ms/step - loss: 1.2459 -
accuracy: 0.6041 - val_loss: 1.1888 - val_accuracy: 0.6114
Epoch 13/50
268/268 [==============================] - 4s 14ms/step - loss: 1.2331 -
accuracy: 0.6086 - val_loss: 1.1765 - val_accuracy: 0.6135
Epoch 14/50
268/268 [==============================] - 3s 12ms/step - loss: 1.2199 -
accuracy: 0.6089 - val_loss: 1.1607 - val_accuracy: 0.6169
Epoch 15/50
268/268 [==============================] - 3s 11ms/step - loss: 1.2084 -
accuracy: 0.6106 - val_loss: 1.1493 - val_accuracy: 0.6243
Epoch 16/50
268/268 [==============================] - 4s 17ms/step - loss: 1.1968 -
accuracy: 0.6125 - val_loss: 1.1377 - val_accuracy: 0.6212
Epoch 17/50
268/268 [==============================] - 4s 16ms/step - loss: 1.1919 -
accuracy: 0.6130 - val_loss: 1.1337 - val_accuracy: 0.6236
Epoch 18/50
268/268 [==============================] - 3s 12ms/step - loss: 1.1778 -
accuracy: 0.6160 - val_loss: 1.1188 - val_accuracy: 0.6294
Epoch 19/50
```

```
268/268 [==============================] - 3s 12ms/step - loss: 1.1737 -
accuracy: 0.6167 - val_loss: 1.1092 - val_accuracy: 0.6327
Epoch 20/50
268/268 [==============================] - 4s 15ms/step - loss: 1.1642 -
accuracy: 0.6217 - val_loss: 1.1023 - val_accuracy: 0.6345
Epoch 21/50
268/268 [==============================] - 5s 17ms/step - loss: 1.1560 -
accuracy: 0.6227 - val_loss: 1.0950 - val_accuracy: 0.6344
Epoch 22/50
268/268 [==============================] - 3s 12ms/step - loss: 1.1482 -
accuracy: 0.6251 - val_loss: 1.0865 - val_accuracy: 0.6371
Epoch 23/50
268/268 [==============================] - 4s 13ms/step - loss: 1.1410 -
accuracy: 0.6274 - val_loss: 1.0774 - val_accuracy: 0.6425
Epoch 24/50
268/268 [==============================] - 4s 15ms/step - loss: 1.1349 -
accuracy: 0.6300 - val_loss: 1.0723 - val_accuracy: 0.6404
Epoch 25/50
268/268 [==============================] - 5s 19ms/step - loss: 1.1291 -
accuracy: 0.6304 - val_loss: 1.0655 - val_accuracy: 0.6423
Epoch 26/50
268/268 [==============================] - 4s 13ms/step - loss: 1.1251 -
accuracy: 0.6303 - val_loss: 1.0587 - val_accuracy: 0.6436
Epoch 27/50
268/268 [==============================] - 3s 12ms/step - loss: 1.1221 -
accuracy: 0.6343 - val_loss: 1.0565 - val_accuracy: 0.6437
Epoch 28/50
268/268 [==============================] - 4s 15ms/step - loss: 1.1125 -
accuracy: 0.6345 - val_loss: 1.0477 - val_accuracy: 0.6448
Epoch 29/50
268/268 [==============================] - 5s 17ms/step - loss: 1.1076 -
accuracy: 0.6365 - val_loss: 1.0426 - val_accuracy: 0.6460
Epoch 30/50
268/268 [==============================] - 3s 12ms/step - loss: 1.1061 -
accuracy: 0.6350 - val_loss: 1.0380 - val_accuracy: 0.6473
Epoch 31/50
268/268 [==============================] - 3s 11ms/step - loss: 1.1012 -
accuracy: 0.6387 - val_loss: 1.0389 - val_accuracy: 0.6462
Epoch 32/50
268/268 [==============================] - 3s 12ms/step - loss: 1.0978 -
accuracy: 0.6388 - val_loss: 1.0351 - val_accuracy: 0.6469
Epoch 33/50
268/268 [==============================] - 5s 20ms/step - loss: 1.0941 -
accuracy: 0.6403 - val_loss: 1.0262 - val_accuracy: 0.6485
Epoch 34/50
268/268 [==============================] - 3s 11ms/step - loss: 1.0944 -
accuracy: 0.6378 - val_loss: 1.0229 - val_accuracy: 0.6480
Epoch 35/50
268/268 [==============================] - 3s 11ms/step - loss: 1.0883 -
accuracy: 0.6395 - val_loss: 1.0204 - val_accuracy: 0.6513
Epoch 36/50
268/268 [==============================] - 3s 11ms/step - loss: 1.0871 -
accuracy: 0.6404 - val_loss: 1.0168 - val_accuracy: 0.6516
Epoch 37/50
268/268 [==============================] - 6s 23ms/step - loss: 1.0848 -
accuracy: 0.6394 - val_loss: 1.0133 - val_accuracy: 0.6530
Epoch 38/50
268/268 [==============================] - 3s 13ms/step - loss: 1.0787 -
accuracy: 0.6414 - val_loss: 1.0112 - val_accuracy: 0.6545
Epoch 39/50
```

```
268/268 [==============================] - 3s 11ms/step - loss: 1.0797 -
accuracy: 0.6406 - val_loss: 1.0090 - val_accuracy: 0.6505
Epoch 40/50
268/268 [==============================] - 3s 12ms/step - loss: 1.0747 -
accuracy: 0.6411 - val_loss: 1.0085 - val_accuracy: 0.6510
Epoch 41/50
268/268 [==============================] - 6s 21ms/step - loss: 1.0705 -
accuracy: 0.6445 - val_loss: 1.0056 - val_accuracy: 0.6583
Epoch 42/50
268/268 [==============================] - 3s 12ms/step - loss: 1.0671 -
accuracy: 0.6431 - val_loss: 1.0008 - val_accuracy: 0.6521
Epoch 43/50
268/268 [==============================] - 3s 11ms/step - loss: 1.0651 -
accuracy: 0.6441 - val_loss: 0.9959 - val_accuracy: 0.6528
Epoch 44/50
268/268 [==============================] - 3s 11ms/step - loss: 1.0654 -
accuracy: 0.6435 - val_loss: 0.9941 - val_accuracy: 0.6604
Epoch 45/50
268/268 [==============================] - 5s 20ms/step - loss: 1.0631 -
accuracy: 0.6455 - val_loss: 0.9911 - val_accuracy: 0.6554
Epoch 46/50
268/268 [==============================] - 3s 12ms/step - loss: 1.0598 -
accuracy: 0.6461 - val_loss: 0.9895 - val_accuracy: 0.6544
Epoch 47/50
268/268 [==============================] - 3s 12ms/step - loss: 1.0542 -
accuracy: 0.6463 - val_loss: 0.9893 - val_accuracy: 0.6548
Epoch 48/50
268/268 [==============================] - 3s 12ms/step - loss: 1.0574 -
accuracy: 0.6470 - val_loss: 0.9844 - val_accuracy: 0.6625
Epoch 49/50
268/268 [==============================] - 5s 19ms/step - loss: 1.0505 -
accuracy: 0.6472 - val_loss: 0.9794 - val_accuracy: 0.6562
Epoch 50/50
268/268 [==============================] - 3s 13ms/step - loss: 1.0463 -
accuracy: 0.6489 - val_loss: 0.9807 - val_accuracy: 0.6586
334/334 [==============================] - 1s 3ms/step - loss: 1.0021 -
accuracy: 0.6556
Test Loss: 1.0021, Test Accuracy: 0.6556
```

65% Accuracy reached after #50 Epoch.



During the training process, the model's performance improves as the number of **epochs** increases. This improvement occurs because each epoch allows the model to iteratively learn from the training data, refining its parameters to better capture the underlying patterns. However, it's important to strike a balance when selecting the number of epochs, as too few epochs may result in underfitting, while too many epochs can lead to overfitting.

an epoch refers to one complete pass through the entire training dataset. During each epoch, the model iterates over the entire dataset, updates its parameters (weights and biases) based on the gradients of the loss function with respect to those parameters, and tries to minimize the loss function.

Cross Validation-10 Folds

```
from sklearn.model_selection import KFold, train_test_split
import numpy as np
import matplotlib.pyplot as plt

kf = KFold(n_splits=10, shuffle=True, random_state=42)

fold_no = 1
acc_per_fold = []
loss_per_fold = []

for train_index, test_index in kf.split(X_seq):
    print(f'Training on fold {fold_no}...')
    X_train, X_test = X_seq[train_index], X_seq[test_index]
    y_train, y_test = y_encoded[train_index], y_encoded[test_index]

    X_train_split, X_val_split, y_train_split, y_val_split =
train_test_split(X_train, y_train, test_size=0.2, random_state=42)

    model = create_model((1, len(api_columns)), num_classes)

    history = model.fit(X_train_split, y_train_split, epochs=10,
batch_size=64, validation_data=(X_val_split, y_val_split))

    loss, accuracy = model.evaluate(X_test, y_test)
    print(f'Fold {fold_no} - Loss: {loss:.4f}, Accuracy:
{accuracy:.4f}')

    acc_per_fold.append(accuracy)
    loss_per_fold.append(loss)

    fold_no += 1

print('Average accuracy: %.4f' % np.mean(acc_per_fold))
print('Average loss: %.4f' % np.mean(loss_per_fold))

# Plotting average accuracy
plt.figure(figsize=(12, 5))
plt.plot(acc_per_fold, marker='o', linestyle='-', color='b')
plt.title('Average Accuracy for each fold')
```

```python
plt.xlabel('Fold')
plt.ylabel('Accuracy')
plt.grid(True)
plt.show()

# Plotting average loss
plt.figure(figsize=(12, 5))
plt.plot(loss_per_fold, marker='o', linestyle='-', color='r')
plt.title('Average Loss for each fold')
plt.xlabel('Fold')
plt.ylabel('Loss')
plt.grid(True)
plt.show()

# Plotting model accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

# Plotting model loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

Output:

```
Training on fold 1...
Epoch 1/10
602/602 [==============================] - 13s 14ms/step - loss: 1.7488 -
accuracy: 0.3553 - val_loss: 1.5714 - val_accuracy: 0.4105
Epoch 2/10
602/602 [==============================] - 7s 11ms/step - loss: 1.5215 -
accuracy: 0.4712 - val_loss: 1.4226 - val_accuracy: 0.5201
Epoch 3/10
602/602 [==============================] - 7s 11ms/step - loss: 1.4204 -
accuracy: 0.5392 - val_loss: 1.3536 - val_accuracy: 0.5606
Epoch 4/10
602/602 [==============================] - 7s 12ms/step - loss: 1.3658 -
accuracy: 0.5732 - val_loss: 1.3064 - val_accuracy: 0.5869
Epoch 5/10
602/602 [==============================] - 9s 14ms/step - loss: 1.3185 -
accuracy: 0.5890 - val_loss: 1.2698 - val_accuracy: 0.5994
Epoch 6/10
602/602 [==============================] - 7s 12ms/step - loss: 1.2831 -
accuracy: 0.5968 - val_loss: 1.2336 - val_accuracy: 0.6029
```

```
Epoch 7/10
602/602 [==============================] - 5s 9ms/step - loss: 1.2589 -
accuracy: 0.6024 - val_loss: 1.2044 - val_accuracy: 0.6078
Epoch 8/10
602/602 [==============================] - 8s 13ms/step - loss: 1.2323 -
accuracy: 0.6066 - val_loss: 1.1823 - val_accuracy: 0.6140
Epoch 9/10
602/602 [==============================] - 5s 9ms/step - loss: 1.2129 -
accuracy: 0.6130 - val_loss: 1.1645 - val_accuracy: 0.6172
Epoch 10/10
602/602 [==============================] - 8s 14ms/step - loss: 1.2022 -
accuracy: 0.6140 - val_loss: 1.1571 - val_accuracy: 0.6144
167/167 [==============================] - 1s 4ms/step - loss: 1.1595 -
accuracy: 0.6254
Fold 1 - Loss: 1.1595, Accuracy: 0.6254
Training on fold 2...
Epoch 1/10
602/602 [==============================] - 11s 11ms/step - loss: 1.7417 -
accuracy: 0.3553 - val_loss: 1.5719 - val_accuracy: 0.3947
Epoch 2/10
602/602 [==============================] - 8s 13ms/step - loss: 1.5190 -
accuracy: 0.4649 - val_loss: 1.4295 - val_accuracy: 0.5225
Epoch 3/10
602/602 [==============================] - 6s 9ms/step - loss: 1.4198 -
accuracy: 0.5374 - val_loss: 1.3559 - val_accuracy: 0.5584
Epoch 4/10
602/602 [==============================] - 8s 14ms/step - loss: 1.3679 -
accuracy: 0.5716 - val_loss: 1.3085 - val_accuracy: 0.5861
Epoch 5/10
602/602 [==============================] - 5s 9ms/step - loss: 1.3228 -
accuracy: 0.5882 - val_loss: 1.2651 - val_accuracy: 0.5937
Epoch 6/10
602/602 [==============================] - 7s 12ms/step - loss: 1.2889 -
accuracy: 0.5976 - val_loss: 1.2344 - val_accuracy: 0.6030
Epoch 7/10
602/602 [==============================] - 6s 10ms/step - loss: 1.2638 -
accuracy: 0.6019 - val_loss: 1.2071 - val_accuracy: 0.6021
Epoch 8/10
602/602 [==============================] - 6s 9ms/step - loss: 1.2360 -
accuracy: 0.6059 - val_loss: 1.1846 - val_accuracy: 0.6110
Epoch 9/10
602/602 [==============================] - 7s 12ms/step - loss: 1.2165 -
accuracy: 0.6103 - val_loss: 1.1672 - val_accuracy: 0.6176
Epoch 10/10
602/602 [==============================] - 6s 9ms/step - loss: 1.2017 -
accuracy: 0.6140 - val_loss: 1.1581 - val_accuracy: 0.6193
167/167 [==============================] - 1s 4ms/step - loss: 1.1508 -
accuracy: 0.6246
Fold 2 - Loss: 1.1508, Accuracy: 0.6246
Training on fold 3...
Epoch 1/10
602/602 [==============================] - 10s 10ms/step - loss: 1.7349 -
accuracy: 0.3661 - val_loss: 1.5659 - val_accuracy: 0.4053
Epoch 2/10
602/602 [==============================] - 7s 12ms/step - loss: 1.5228 -
accuracy: 0.4661 - val_loss: 1.4293 - val_accuracy: 0.5113
Epoch 3/10
602/602 [==============================] - 5s 8ms/step - loss: 1.4216 -
accuracy: 0.5373 - val_loss: 1.3539 - val_accuracy: 0.5649
Epoch 4/10
```

```
602/602 [==============================] - 7s 12ms/step - loss: 1.3638 -
accuracy: 0.5690 - val_loss: 1.3016 - val_accuracy: 0.5867
Epoch 5/10
602/602 [==============================] - 6s 9ms/step - loss: 1.3184 -
accuracy: 0.5871 - val_loss: 1.2574 - val_accuracy: 0.5981
Epoch 6/10
602/602 [==============================] - 5s 9ms/step - loss: 1.2813 -
accuracy: 0.5972 - val_loss: 1.2292 - val_accuracy: 0.6006
Epoch 7/10
602/602 [==============================] - 7s 12ms/step - loss: 1.2539 -
accuracy: 0.6033 - val_loss: 1.2016 - val_accuracy: 0.6085
Epoch 8/10
602/602 [==============================] - 5s 9ms/step - loss: 1.2298 -
accuracy: 0.6055 - val_loss: 1.1792 - val_accuracy: 0.6127
Epoch 9/10
602/602 [==============================] - 8s 13ms/step - loss: 1.2119 -
accuracy: 0.6105 - val_loss: 1.1646 - val_accuracy: 0.6202
Epoch 10/10
602/602 [==============================] - 5s 9ms/step - loss: 1.1987 -
accuracy: 0.6136 - val_loss: 1.1499 - val_accuracy: 0.6190
167/167 [==============================] - 1s 4ms/step - loss: 1.1625 -
accuracy: 0.6155
Fold 3 - Loss: 1.1625, Accuracy: 0.6155
Training on fold 4...
Epoch 1/10
602/602 [==============================] - 10s 11ms/step - loss: 1.7357 -
accuracy: 0.3615 - val_loss: 1.5665 - val_accuracy: 0.3966
Epoch 2/10
602/602 [==============================] - 7s 12ms/step - loss: 1.5181 -
accuracy: 0.4740 - val_loss: 1.4219 - val_accuracy: 0.5178
Epoch 3/10
602/602 [==============================] - 5s 9ms/step - loss: 1.4265 -
accuracy: 0.5396 - val_loss: 1.3539 - val_accuracy: 0.5605
Epoch 4/10
602/602 [==============================] - 8s 13ms/step - loss: 1.3701 -
accuracy: 0.5710 - val_loss: 1.3064 - val_accuracy: 0.5863
Epoch 5/10
602/602 [==============================] - 6s 9ms/step - loss: 1.3233 -
accuracy: 0.5882 - val_loss: 1.2648 - val_accuracy: 0.5974
Epoch 6/10
602/602 [==============================] - 7s 12ms/step - loss: 1.2845 -
accuracy: 0.5967 - val_loss: 1.2324 - val_accuracy: 0.6023
Epoch 7/10
602/602 [==============================] - 6s 11ms/step - loss: 1.2603 -
accuracy: 0.6002 - val_loss: 1.2090 - val_accuracy: 0.6053
Epoch 8/10
602/602 [==============================] - 6s 9ms/step - loss: 1.2340 -
accuracy: 0.6059 - val_loss: 1.1917 - val_accuracy: 0.6120
Epoch 9/10
602/602 [==============================] - 8s 13ms/step - loss: 1.2181 -
accuracy: 0.6089 - val_loss: 1.1766 - val_accuracy: 0.6149
Epoch 10/10
602/602 [==============================] - 5s 9ms/step - loss: 1.2014 -
accuracy: 0.6133 - val_loss: 1.1600 - val_accuracy: 0.6158
167/167 [==============================] - 0s 3ms/step - loss: 1.1373 -
accuracy: 0.6231
Fold 4 - Loss: 1.1373, Accuracy: 0.6231
Training on fold 5...
Epoch 1/10
602/602 [==============================] - 12s 11ms/step - loss: 1.7442 -
accuracy: 0.3574 - val_loss: 1.5762 - val_accuracy: 0.4139
```

```
Epoch 2/10
602/602 [==============================] - 7s 11ms/step - loss: 1.5194 -
accuracy: 0.4693 - val_loss: 1.4322 - val_accuracy: 0.5240
Epoch 3/10
602/602 [==============================] - 7s 11ms/step - loss: 1.4188 -
accuracy: 0.5418 - val_loss: 1.3619 - val_accuracy: 0.5661
Epoch 4/10
602/602 [==============================] - 5s 8ms/step - loss: 1.3644 -
accuracy: 0.5706 - val_loss: 1.3146 - val_accuracy: 0.5810
Epoch 5/10
602/602 [==============================] - 8s 13ms/step - loss: 1.3214 -
accuracy: 0.5874 - val_loss: 1.2721 - val_accuracy: 0.5942
Epoch 6/10
602/602 [==============================] - 5s 9ms/step - loss: 1.2873 -
accuracy: 0.5949 - val_loss: 1.2409 - val_accuracy: 0.5992
Epoch 7/10
602/602 [==============================] - 6s 11ms/step - loss: 1.2622 -
accuracy: 0.6006 - val_loss: 1.2136 - val_accuracy: 0.6074
Epoch 8/10
602/602 [==============================] - 6s 11ms/step - loss: 1.2381 -
accuracy: 0.6051 - val_loss: 1.1934 - val_accuracy: 0.6116
Epoch 9/10
602/602 [==============================] - 5s 9ms/step - loss: 1.2173 -
accuracy: 0.6067 - val_loss: 1.1739 - val_accuracy: 0.6136
Epoch 10/10
602/602 [==============================] - 8s 13ms/step - loss: 1.2033 -
accuracy: 0.6130 - val_loss: 1.1618 - val_accuracy: 0.6162
167/167 [==============================] - 0s 3ms/step - loss: 1.1485 -
accuracy: 0.6278
Fold 5 - Loss: 1.1485, Accuracy: 0.6278
Training on fold 6...
Epoch 1/10
602/602 [==============================] - 13s 12ms/step - loss: 1.7392 -
accuracy: 0.3570 - val_loss: 1.5553 - val_accuracy: 0.4189
Epoch 2/10
602/602 [==============================] - 8s 13ms/step - loss: 1.5068 -
accuracy: 0.4788 - val_loss: 1.4088 - val_accuracy: 0.5271
Epoch 3/10
602/602 [==============================] - 6s 10ms/step - loss: 1.4123 -
accuracy: 0.5456 - val_loss: 1.3491 - val_accuracy: 0.5625
Epoch 4/10
602/602 [==============================] - 7s 11ms/step - loss: 1.3559 -
accuracy: 0.5758 - val_loss: 1.2938 - val_accuracy: 0.5846
Epoch 5/10
602/602 [==============================] - 7s 12ms/step - loss: 1.3110 -
accuracy: 0.5909 - val_loss: 1.2505 - val_accuracy: 0.5984
Epoch 6/10
602/602 [==============================] - 6s 10ms/step - loss: 1.2782 -
accuracy: 0.5987 - val_loss: 1.2196 - val_accuracy: 0.6020
Epoch 7/10
602/602 [==============================] - 8s 14ms/step - loss: 1.2509 -
accuracy: 0.6061 - val_loss: 1.1979 - val_accuracy: 0.6064
Epoch 8/10
602/602 [==============================] - 6s 10ms/step - loss: 1.2279 -
accuracy: 0.6091 - val_loss: 1.1768 - val_accuracy: 0.6088
Epoch 9/10
602/602 [==============================] - 8s 14ms/step - loss: 1.2111 -
accuracy: 0.6117 - val_loss: 1.1630 - val_accuracy: 0.6152
Epoch 10/10
602/602 [==============================] - 6s 10ms/step - loss: 1.1935 -
accuracy: 0.6153 - val_loss: 1.1481 - val_accuracy: 0.6190
```

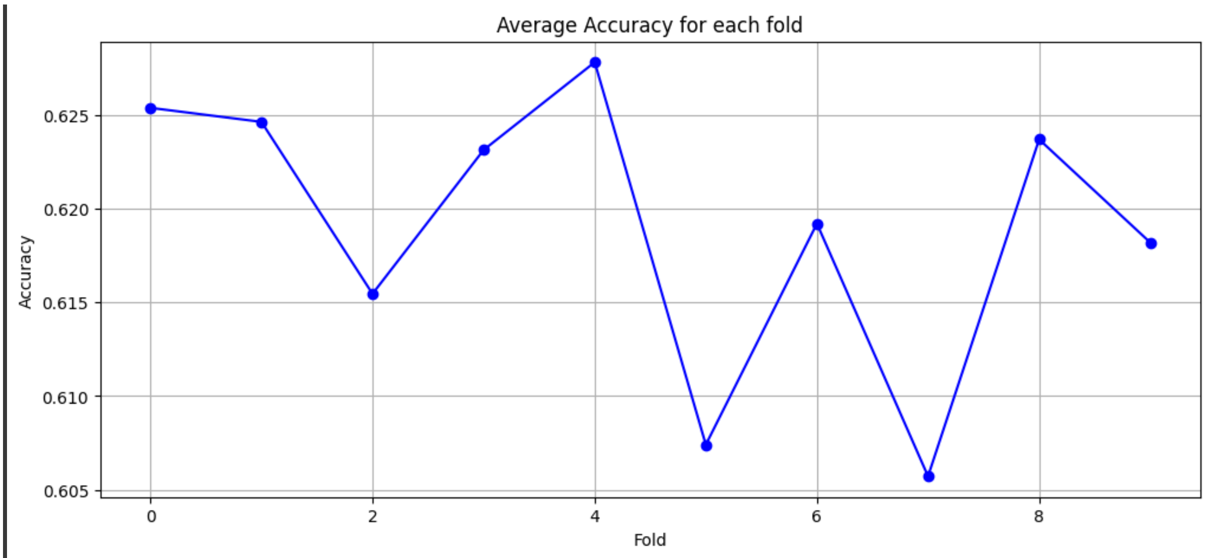
```
167/167 [==============================] - 1s 3ms/step - loss: 1.1866 -
accuracy: 0.6074
Fold 6 - Loss: 1.1866, Accuracy: 0.6074
Training on fold 7...
Epoch 1/10
602/602 [==============================] - 12s 12ms/step - loss: 1.7608 -
accuracy: 0.3438 - val_loss: 1.5859 - val_accuracy: 0.4083
Epoch 2/10
602/602 [==============================] - 6s 11ms/step - loss: 1.5313 -
accuracy: 0.4633 - val_loss: 1.4340 - val_accuracy: 0.5068
Epoch 3/10
602/602 [==============================] - 7s 12ms/step - loss: 1.4278 -
accuracy: 0.5355 - val_loss: 1.3611 - val_accuracy: 0.5575
Epoch 4/10
602/602 [==============================] - 6s 9ms/step - loss: 1.3709 -
accuracy: 0.5681 - val_loss: 1.3110 - val_accuracy: 0.5873
Epoch 5/10
602/602 [==============================] - 8s 14ms/step - loss: 1.3167 -
accuracy: 0.5891 - val_loss: 1.2736 - val_accuracy: 0.5969
Epoch 6/10
602/602 [==============================] - 6s 10ms/step - loss: 1.2836 -
accuracy: 0.5993 - val_loss: 1.2372 - val_accuracy: 0.6056
Epoch 7/10
602/602 [==============================] - 8s 13ms/step - loss: 1.2554 -
accuracy: 0.6032 - val_loss: 1.2140 - val_accuracy: 0.6066
Epoch 8/10
602/602 [==============================] - 6s 9ms/step - loss: 1.2333 -
accuracy: 0.6076 - val_loss: 1.1892 - val_accuracy: 0.6169
Epoch 9/10
602/602 [==============================] - 6s 10ms/step - loss: 1.2166 -
accuracy: 0.6098 - val_loss: 1.1799 - val_accuracy: 0.6197
Epoch 10/10
602/602 [==============================] - 7s 12ms/step - loss: 1.1998 -
accuracy: 0.6133 - val_loss: 1.1568 - val_accuracy: 0.6241
167/167 [==============================] - 0s 3ms/step - loss: 1.1623 -
accuracy: 0.6192
Fold 7 - Loss: 1.1623, Accuracy: 0.6192
Training on fold 8...
Epoch 1/10
602/602 [==============================] - 13s 15ms/step - loss: 1.7404 -
accuracy: 0.3592 - val_loss: 1.5739 - val_accuracy: 0.3960
Epoch 2/10
602/602 [==============================] - 6s 9ms/step - loss: 1.5269 -
accuracy: 0.4610 - val_loss: 1.4385 - val_accuracy: 0.5112
Epoch 3/10
602/602 [==============================] - 8s 13ms/step - loss: 1.4208 -
accuracy: 0.5383 - val_loss: 1.3667 - val_accuracy: 0.5540
Epoch 4/10
602/602 [==============================] - 6s 9ms/step - loss: 1.3641 -
accuracy: 0.5696 - val_loss: 1.3223 - val_accuracy: 0.5796
Epoch 5/10
602/602 [==============================] - 7s 12ms/step - loss: 1.3198 -
accuracy: 0.5876 - val_loss: 1.2800 - val_accuracy: 0.5985
Epoch 6/10
602/602 [==============================] - 6s 11ms/step - loss: 1.2837 -
accuracy: 0.5969 - val_loss: 1.2430 - val_accuracy: 0.6037
Epoch 7/10
602/602 [==============================] - 6s 9ms/step - loss: 1.2553 -
accuracy: 0.6026 - val_loss: 1.2209 - val_accuracy: 0.6071
Epoch 8/10
```

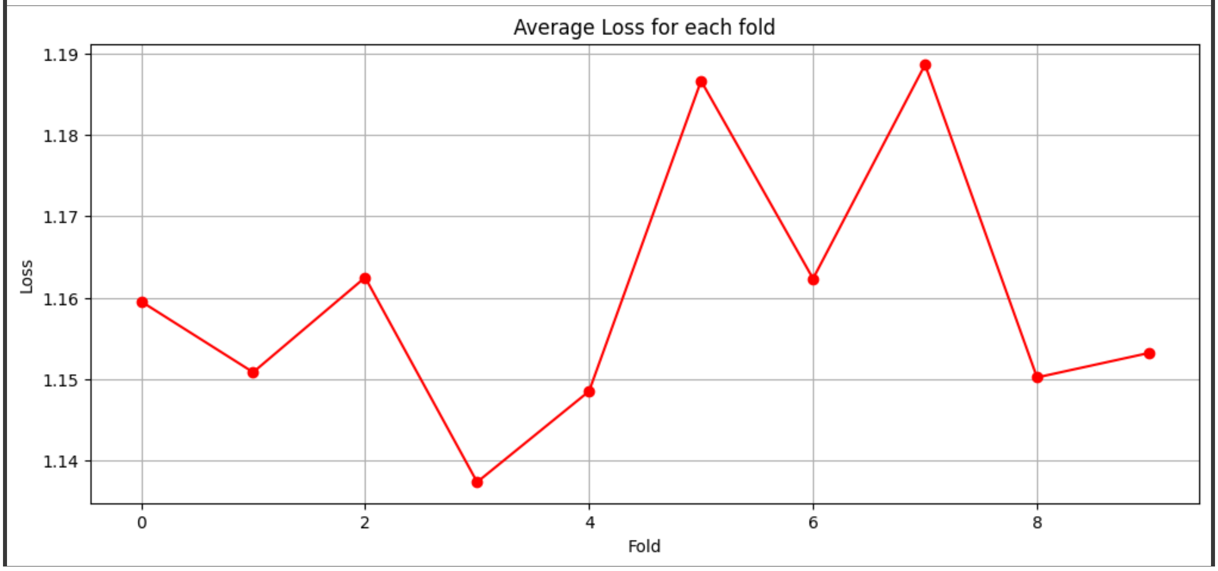
```
602/602 [==============================] - 8s 13ms/step - loss: 1.2292 -
accuracy: 0.6096 - val_loss: 1.2003 - val_accuracy: 0.6086
Epoch 9/10
602/602 [==============================] - 6s 9ms/step - loss: 1.2092 -
accuracy: 0.6105 - val_loss: 1.1803 - val_accuracy: 0.6128
Epoch 10/10
602/602 [==============================] - 8s 13ms/step - loss: 1.1940 -
accuracy: 0.6147 - val_loss: 1.1707 - val_accuracy: 0.6141
167/167 [==============================] - 0s 3ms/step - loss: 1.1886 -
accuracy: 0.6057
Fold 8 - Loss: 1.1886, Accuracy: 0.6057
Training on fold 9...
Epoch 1/10
602/602 [==============================] - 12s 13ms/step - loss: 1.7347 -
accuracy: 0.3649 - val_loss: 1.5725 - val_accuracy: 0.4060
Epoch 2/10
602/602 [==============================] - 6s 9ms/step - loss: 1.5229 -
accuracy: 0.4619 - val_loss: 1.4380 - val_accuracy: 0.5224
Epoch 3/10
602/602 [==============================] - 6s 11ms/step - loss: 1.4235 -
accuracy: 0.5382 - val_loss: 1.3588 - val_accuracy: 0.5575
Epoch 4/10
602/602 [==============================] - 7s 12ms/step - loss: 1.3634 -
accuracy: 0.5710 - val_loss: 1.3100 - val_accuracy: 0.5863
Epoch 5/10
602/602 [==============================] - 5s 9ms/step - loss: 1.3206 -
accuracy: 0.5855 - val_loss: 1.2704 - val_accuracy: 0.5948
Epoch 6/10
602/602 [==============================] - 8s 13ms/step - loss: 1.2862 -
accuracy: 0.5956 - val_loss: 1.2373 - val_accuracy: 0.6040
Epoch 7/10
602/602 [==============================] - 6s 9ms/step - loss: 1.2573 -
accuracy: 0.6026 - val_loss: 1.2136 - val_accuracy: 0.6087
Epoch 8/10
602/602 [==============================] - 8s 13ms/step - loss: 1.2371 -
accuracy: 0.6074 - val_loss: 1.1986 - val_accuracy: 0.6135
Epoch 9/10
602/602 [==============================] - 6s 9ms/step - loss: 1.2167 -
accuracy: 0.6107 - val_loss: 1.1757 - val_accuracy: 0.6183
Epoch 10/10
602/602 [==============================] - 7s 11ms/step - loss: 1.1988 -
accuracy: 0.6158 - val_loss: 1.1624 - val_accuracy: 0.6211
167/167 [==============================] - 0s 3ms/step - loss: 1.1502 -
accuracy: 0.6237
Fold 9 - Loss: 1.1502, Accuracy: 0.6237
Training on fold 10...
Epoch 1/10
602/602 [==============================] - 11s 13ms/step - loss: 1.7482 -
accuracy: 0.3545 - val_loss: 1.5620 - val_accuracy: 0.3938
Epoch 2/10
602/602 [==============================] - 7s 11ms/step - loss: 1.5155 -
accuracy: 0.4678 - val_loss: 1.4096 - val_accuracy: 0.5225
Epoch 3/10
602/602 [==============================] - 6s 9ms/step - loss: 1.4228 -
accuracy: 0.5412 - val_loss: 1.3399 - val_accuracy: 0.5764
Epoch 4/10
602/602 [==============================] - 8s 13ms/step - loss: 1.3657 -
accuracy: 0.5701 - val_loss: 1.2918 - val_accuracy: 0.5896
Epoch 5/10
602/602 [==============================] - 6s 9ms/step - loss: 1.3186 -
accuracy: 0.5864 - val_loss: 1.2468 - val_accuracy: 0.6078
```

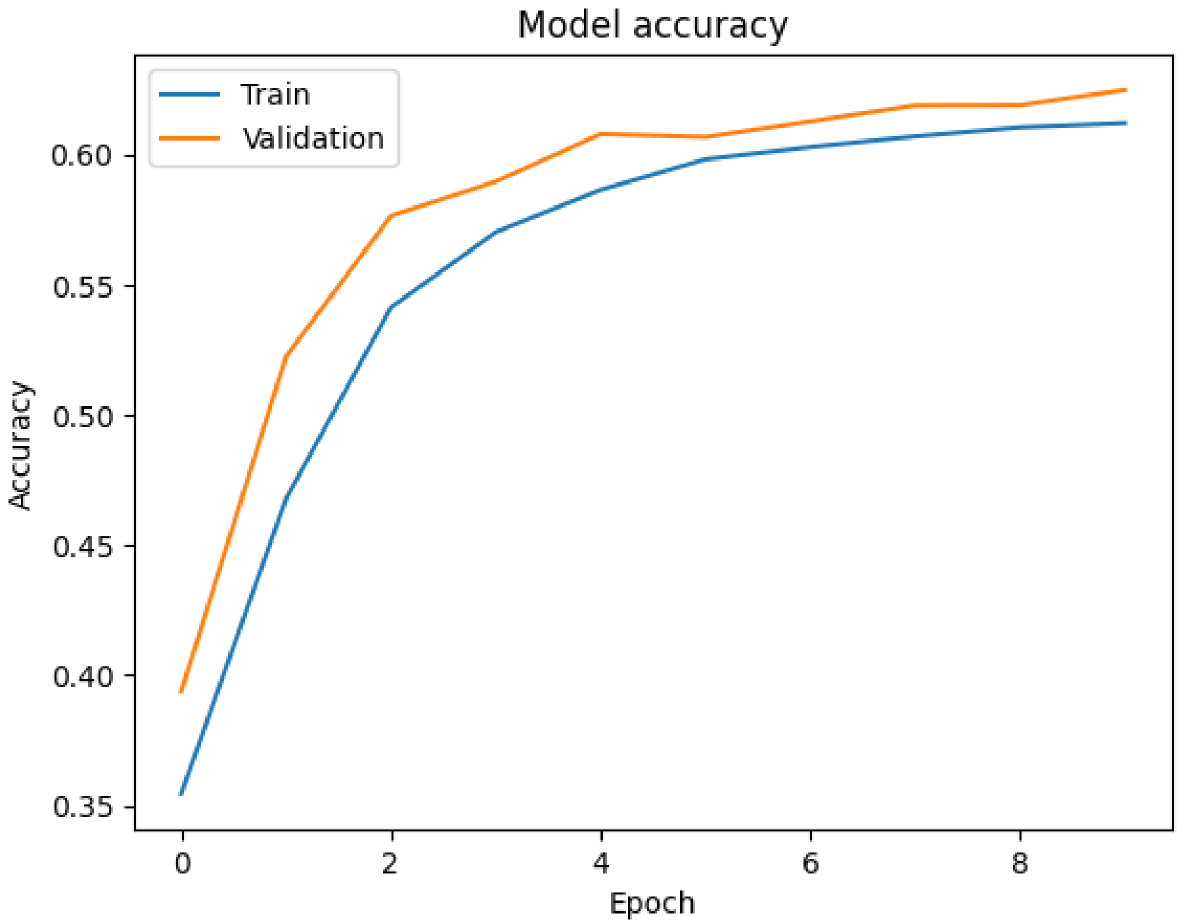
```
Epoch 6/10
602/602 [==============================] - 8s 13ms/step - loss: 1.2812 -
accuracy: 0.5981 - val_loss: 1.2187 - val_accuracy: 0.6067
Epoch 7/10
602/602 [==============================] - 6s 9ms/step - loss: 1.2545 -
accuracy: 0.6028 - val_loss: 1.1927 - val_accuracy: 0.6127
Epoch 8/10
602/602 [==============================] - 6s 10ms/step - loss: 1.2325 -
accuracy: 0.6069 - val_loss: 1.1700 - val_accuracy: 0.6188
Epoch 9/10
602/602 [==============================] - 7s 12ms/step - loss: 1.2157 -
accuracy: 0.6104 - val_loss: 1.1585 - val_accuracy: 0.6189
Epoch 10/10
602/602 [==============================] - 5s 9ms/step - loss: 1.1989 -
accuracy: 0.6120 - val_loss: 1.1413 - val_accuracy: 0.6247
167/167 [==============================] - 1s 5ms/step - loss: 1.1532 -
accuracy: 0.6182
Fold 10 - Loss: 1.1532, Accuracy: 0.6182
```

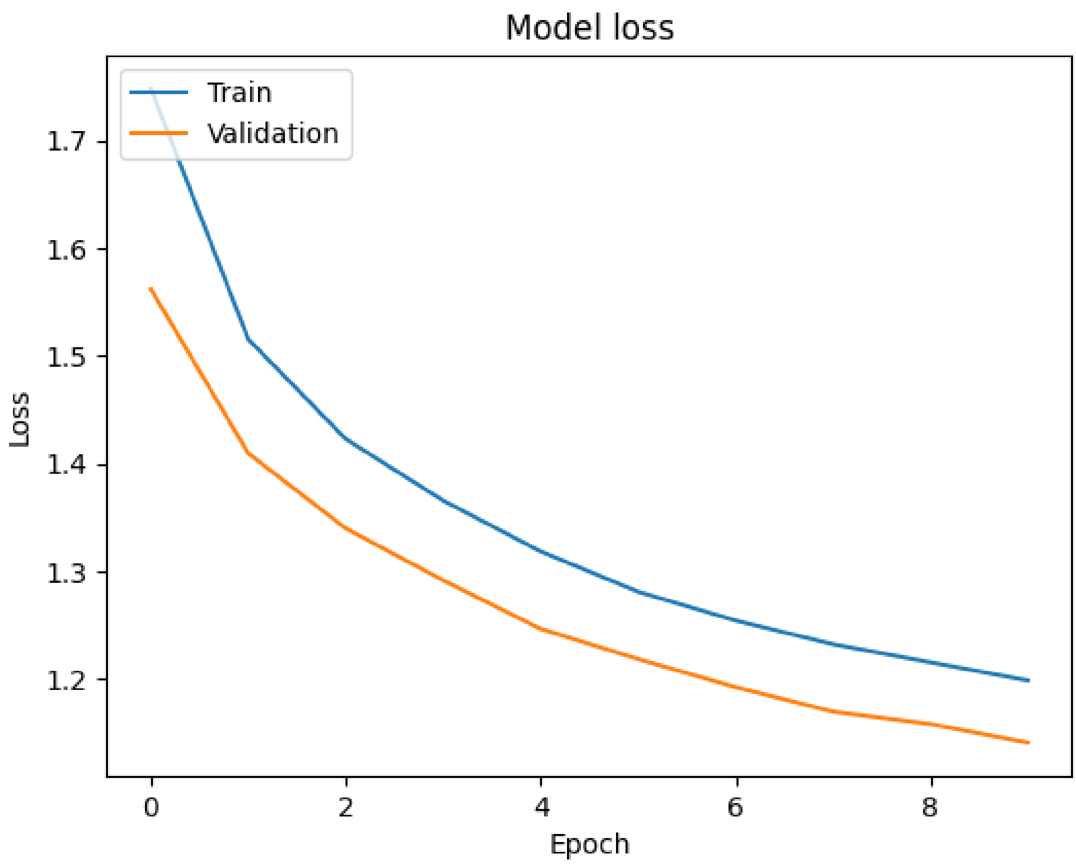
Average accuracy: 61.91%

Average loss: 1.1600






Model accuracy
Train
Validation
Accuracy
0.60
0.55
0.50
0.45
0.40
0.35
0
2
4
6
8
Epoch


Model loss
Train
Validation
Loss
1.7
1.6
1.5
1.4
1.3
1.2
0
2
4
6
8
Epoch