In [ ]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, losses
import matplotlib.pyplot as plt
import os
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_scor
```

For this section, we'll design a Neural Network to predict the genre of a film based off of numerical values.

In [ ]:
```
df = pd.read_csv("./data/nan_removed_cleaned_data.csv")

# Convert our datetime objects to Unix timestamps
df['release_unix'] = pd.to_datetime(df['release_date'])
df['release_unix'] = df.release_unix.values.astype(np.int64) // 10 ** 9

# This will set the first_genre to the first genre that appears in the list of genr
df['first_genre'] = df.apply(lambda row: row['genres'].strip('[]').replace("'", '')

# Remove any rows that have 'None' as their genre. We will regard these as missing
df = df[df['first_genre'] != 'None']
```

Now, let's cut our problem down to just 3 genres, since the other genres are a bit too obscure according to this data:

In [ ]:
```
df['first_genre'].value_counts()
```

Out[ ]:
```
Drama             2969
Comedy            2315
Horror            1673
Documentary       1541
Action            1196
Thriller           622
Crime              454
Adventure          411
ScienceFiction     357
Animation          290
Romance            253
Music              224
Fantasy            221
Family             191
Mystery            141
Western            132
War                 67
History             61
TVMovie             49
Name: first_genre, dtype: int64
```

In [ ]:
```
df = df[(df['first_genre'] == 'Drama') | (df['first_genre'] == 'Comedy') | (df['fir
```

```
# Extract the unique genres
unique_genres = df['first_genre'].unique()
# Assign an integer to each genre
genre_dict = {}
for idx, i in enumerate(unique_genres):
    genre_dict[i] = idx

df['genre_int'] = df.apply(lambda row: genre_dict[row['first_genre']], axis=1)

display(df['first_genre'].value_counts())

df[['genre_int', 'first_genre']].sample(10)
```

```
Drama           2969
Comedy          2315
Documentary     1541
Name: first_genre, dtype: int64
```

Out[ ]:

	genre_int	first_genre
10335	0	Documentary
7845	1	Drama
13899	2	Comedy
534	2	Comedy
7574	2	Comedy
11960	1	Drama
12773	2	Comedy
14339	2	Comedy
5753	1	Drama
6910	1	Drama

Great! Now we have integer labels to predict on.

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['budget', 'genres', 'id', 'imdb_id', 'original_title', 'overview',
       'popularity', 'poster_path', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'tagline', 'title', 'vote_average', 'vote_count',
       'log_popularity', 'title_length', 'num_languages', 'num_genres',
       'imdb_rating', 'imdb_budget', 'imdb_revenue', 'budget_currency',
       'revenue_currency', 'converted_budget', 'converted_revenue',
       'combined_budget', 'combined_revenue', 'release_unix', 'first_genre',
       'genre_int'],
      dtype='object')
```

Now, we want to only extract the columns that will be relevant to our Neural Network prediction.

Namely, numeric values that aren't unimportant (i.e. not id, etc.)

In [ ]:
```
df = df[['runtime', 'vote_average', 'vote_count', 'log_popularity', 'title_length',
         'num_genres', 'imdb_rating', 'combined_budget', 'combined_revenue', 'rele

# Next, remove any missing data
df.dropna(inplace=True)
```

Now that we have our finalized dataset, we can separate it into training and testing.

In [ ]:
```
# Separate into train and test
train = df.sample(frac=0.8,random_state=1612)
test = df.drop(train.index)

display(train.head(5))
display(test.head(5))

# Extract y vs. x
train_y = train['genre_int'].to_numpy()
test_y = test['genre_int'].to_numpy()
train_x = train.drop('genre_int', axis=1).to_numpy()
test_x = test.drop('genre_int', axis=1).to_numpy()

display(np.unique(train_y, return_counts=True))
display(np.unique(test_y, return_counts=True))

display(train_x.shape)
```

	runtime	vote_average	vote_count	log_popularity	title_length	num_languages	num_genr
9416	60	6.5	7	0.518794	8	1	
5504	72	6.4	28	0.898534	10	2	
10094	89	3.7	7	1.094604	17	1	
8196	77	6.0	1	0.470004	11	1	
939	114	5.2	353	2.077440	8	1	

	runtime	vote_average	vote_count	log_popularity	title_length	num_languages	num_genres
7	91	7.400	88	2.077690	18	1	1
17	110	7.588	3407	2.999525	12	1	3
34	133	7.780	618	2.982495	30	4	2
50	130	7.468	2579	2.898450	20	4	3
55	89	6.600	21	1.356608	14	1	1

```
(array([0, 1, 2], dtype=int64), array([1017, 2154, 1729], dtype=int64))
(array([0, 1, 2], dtype=int64), array([245, 547, 433], dtype=int64))
(4900, 11)
```

Awesome! Now we have separation, **and** both our train and test sets have examples of each genre.

Notice that we have 5 classes.

Now that we have all of our data, we can set up a Neural Network to train and (hopefully) predict between these 5 classes.

First, set up our various training-specific variables:

In [ ]:
```
# Activation function to use for our inner layers
act = 'relu'

# Learning rate
lr = 0.0005

# Number of epochs
epochs = 100

# Loss function - Here we're using a modified version
# of cross-entropy for multi-class classification.
loss_fun = losses.SparseCategoricalCrossentropy(from_logits=True)

# Number of prediction classes:
n_classes = len(np.unique(train_y))
```

Next, we notice from previously that the amount of samples of each class is quite skewed. This will significantly decrease the performance of our model, so TensorFlow lets us assign class weights to each class.

(see https://www.tensorflow.org/tutorials/structured_data/imbalanced_data#calculate_class_weights)

In [ ]:
```
_, counts = np.unique(train_y, return_counts=True)
class_weights = {}
for idx, i in enumerate(counts):
    class_weights[idx] = (1 / i) * (train_y.shape[0] / n_classes)

display(class_weights)
```

```
{0: 1.6060308095706326, 1: 0.7582791705354379, 2: 0.9446693657219972}
```

In [ ]:
```
# Setting up our network for multi-class classification

model = keras.Sequential([
    # Input layer - takes in the 12 predictors
    layers.InputLayer(input_shape=(train_x.shape[1],)),
    # Normalize our inputs
    #layers.Normalization(axis=-1),
    # 2 Hidden layers, each with 32 units
    layers.Dense(32, activation=act),
    layers.Dense(64, activation=act),
    layers.Dense(128, activation=act),
    layers.Dense(64, activation=act),
    layers.Dense(32, activation=act),
    # Predict one of the classes
```

```
    layers.Dense(n_classes)])

# Print a summary of the model
model.summary()

# Compile, using the Adam optimizer for learning rate
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=lr),
                loss=loss_fun, metrics=['accuracy'])

'''
The following is TensorFlow-specific.
This just sets up a checkpointer to save our progress throughout the training.
'''

# Set up the checkpointing
checkpoint_path = "training/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)

# Create a callback that saves the model's weights
cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_path,
                                                 save_weights_only=True,
                                                 verbose=1)

'''
END
'''

# Train the model
history = model.fit(train_x, train_y, epochs=epochs, batch_size=8,
                    validation_split=0.25, callbacks=[cp_callback], class_weight=cl

def plot_loss(history):
    plt.plot(history.history['loss'], label='loss')
    plt.plot(history.history['val_loss'], label='val_loss')
    plt.xlabel('Epoch')
    plt.ylabel('Error (Categorical Cross-Entropy)')
    plt.title('Cross-Entropy for Film Genre Classification')
    plt.legend()
    plt.grid(True)
    plt.savefig('./imgs/nn_ims/training_loss.png')
    plt.show()

# Plot the loss
plot_loss(history)

# Evaluate on the withheld test data
test_result = model.evaluate(test_x, test_y)
print(test_result)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                384

 dense_1 (Dense)             (None, 64)                2112

 dense_2 (Dense)             (None, 128)               8320

 dense_3 (Dense)             (None, 64)                8256

 dense_4 (Dense)             (None, 32)                2080

 dense_5 (Dense)             (None, 3)                 99

=================================================================
Total params: 21,251
Trainable params: 21,251
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
437/460 [===========================>..] - ETA: 0s - loss: 3376482.7500 - accurac
y: 0.3493
Epoch 1: saving model to training\cp.ckpt
460/460 [==============================] - 2s 3ms/step - loss: 3362815.0000 - accu
racy: 0.3483 - val_loss: 3951294.5000 - val_accuracy: 0.2359
Epoch 2/100
436/460 [===========================>..] - ETA: 0s - loss: 1695141.3750 - accurac
y: 0.3615
Epoch 2: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1672640.7500 - accu
racy: 0.3584 - val_loss: 1310934.6250 - val_accuracy: 0.3698
Epoch 3/100
456/460 [============================>.] - ETA: 0s - loss: 1282534.7500 - accurac
y: 0.3544
Epoch 3: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1286010.7500 - accu
racy: 0.3535 - val_loss: 1511550.0000 - val_accuracy: 0.3135
Epoch 4/100
453/460 [============================>.] - ETA: 0s - loss: 581704.7500 - accuracy:
0.3645
Epoch 4: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 580584.0625 - accur
acy: 0.3641 - val_loss: 215305.7500 - val_accuracy: 0.3494
Epoch 5/100
455/460 [============================>.] - ETA: 0s - loss: 642363.3125 - accuracy:
0.3692
Epoch 5: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 642286.4375 - accur
acy: 0.3676 - val_loss: 345345.8125 - val_accuracy: 0.3200
Epoch 6/100
431/460 [===========================>..] - ETA: 0s - loss: 315931.1875 - accuracy:
0.3643
Epoch 6: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 304012.1250 - accur
```

```
acy: 0.3652 - val_loss: 295854.1250 - val_accuracy: 0.3494
Epoch 7/100
458/460 [============================>.] - ETA: 0s - loss: 283865.5000 - accuracy:
0.3687
Epoch 7: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 283361.8438 - accur
acy: 0.3687 - val_loss: 293364.2812 - val_accuracy: 0.3690
Epoch 8/100
459/460 [============================>.] - ETA: 0s - loss: 210717.8906 - accuracy:
0.3627
Epoch 8: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 210549.4062 - accur
acy: 0.3630 - val_loss: 165553.2188 - val_accuracy: 0.4155
Epoch 9/100
441/460 [===========================>..] - ETA: 0s - loss: 174026.9062 - accuracy:
0.3829
Epoch 9: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 180727.5469 - accur
acy: 0.3829 - val_loss: 269145.0625 - val_accuracy: 0.3241
Epoch 10/100
454/460 [============================>.] - ETA: 0s - loss: 173467.9062 - accuracy:
0.3645
Epoch 10: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 174344.8438 - accur
acy: 0.3635 - val_loss: 293871.8438 - val_accuracy: 0.4351
Epoch 11/100
431/460 [===========================>..] - ETA: 0s - loss: 154913.7969 - accuracy:
0.3619
Epoch 11: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 153819.9062 - accur
acy: 0.3616 - val_loss: 106963.4141 - val_accuracy: 0.3682
Epoch 12/100
448/460 [============================>.] - ETA: 0s - loss: 125204.9375 - accuracy:
0.3672
Epoch 12: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 124821.5312 - accur
acy: 0.3679 - val_loss: 105008.1953 - val_accuracy: 0.3918
Epoch 13/100
443/460 [===========================>..] - ETA: 0s - loss: 88478.3984 - accuracy:
0.3758
Epoch 13: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 86687.2422 - accura
cy: 0.3750 - val_loss: 55214.1055 - val_accuracy: 0.3486
Epoch 14/100
454/460 [============================>.] - ETA: 0s - loss: 66798.0312 - accuracy:
0.3833
Epoch 14: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 66214.2109 - accura
cy: 0.3837 - val_loss: 16189.0771 - val_accuracy: 0.4441
Epoch 15/100
439/460 [===========================>..] - ETA: 0s - loss: 46708.2383 - accuracy:
0.3781
Epoch 15: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 46749.7266 - accura
cy: 0.3766 - val_loss: 17223.5449 - val_accuracy: 0.3690
Epoch 16/100
```

```
445/460 [============================>.] - ETA: 0s - loss: 59132.2656 - accuracy:
0.3801
Epoch 16: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 59616.6875 - accura
cy: 0.3807 - val_loss: 71996.9141 - val_accuracy: 0.3576
Epoch 17/100
446/460 [============================>.] - ETA: 0s - loss: 44328.8828 - accuracy:
0.3845
Epoch 17: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 46388.9297 - accura
cy: 0.3823 - val_loss: 98474.3438 - val_accuracy: 0.2482
Epoch 18/100
454/460 [============================>.] - ETA: 0s - loss: 45088.5898 - accuracy:
0.3593
Epoch 18: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 45079.1523 - accura
cy: 0.3586 - val_loss: 30983.3379 - val_accuracy: 0.3992
Epoch 19/100
455/460 [============================>.] - ETA: 0s - loss: 36702.1875 - accuracy:
0.3698
Epoch 19: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 36707.0391 - accura
cy: 0.3695 - val_loss: 47422.1094 - val_accuracy: 0.3698
Epoch 20/100
451/460 [============================>.] - ETA: 0s - loss: 49514.8789 - accuracy:
0.3756
Epoch 20: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 49103.2773 - accura
cy: 0.3755 - val_loss: 5954.2715 - val_accuracy: 0.3004
Epoch 21/100
453/460 [============================>.] - ETA: 0s - loss: 521.1002 - accuracy: 0.
3590
Epoch 21: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 516.4199 - accurac
y: 0.3578 - val_loss: 1081.4913 - val_accuracy: 0.3502
Epoch 22/100
455/460 [============================>.] - ETA: 0s - loss: 89.6682 - accuracy: 0.3
591
Epoch 22: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 88.8253 - accuracy:
0.3586 - val_loss: 648.4824 - val_accuracy: 0.3527
Epoch 23/100
453/460 [============================>.] - ETA: 0s - loss: 82.4045 - accuracy: 0.3
587
Epoch 23: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 81.2761 - accuracy:
0.3581 - val_loss: 371.4450 - val_accuracy: 0.3510
Epoch 24/100
449/460 [============================>.] - ETA: 0s - loss: 224.2369 - accuracy: 0.
3555
Epoch 24: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 325.0161 - accurac
y: 0.3562 - val_loss: 1442.0295 - val_accuracy: 0.3510
Epoch 25/100
456/460 [============================>.] - ETA: 0s - loss: 783.6525 - accuracy: 0.
3580
```

```
Epoch 25: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 777.9032 - accurac
y: 0.3578 - val_loss: 556.4061 - val_accuracy: 0.3502
Epoch 26/100
436/460 [===========================>..] - ETA: 0s - loss: 15.5558 - accuracy: 0.3
564
Epoch 26: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 14.8395 - accuracy:
0.3559 - val_loss: 684.4354 - val_accuracy: 0.3502
Epoch 27/100
435/460 [===========================>..] - ETA: 0s - loss: 50.9582 - accuracy: 0.3
589
Epoch 27: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 48.3275 - accuracy:
0.3570 - val_loss: 668.8493 - val_accuracy: 0.3510
Epoch 28/100
451/460 [============================>.] - ETA: 0s - loss: 85.5434 - accuracy: 0.3
564
Epoch 28: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 84.0046 - accuracy:
0.3562 - val_loss: 547.2642 - val_accuracy: 0.3494
Epoch 29/100
448/460 [============================>.] - ETA: 0s - loss: 1.1022 - accuracy: 0.38
59
Epoch 29: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.1033 - accuracy:
0.3831 - val_loss: 547.7786 - val_accuracy: 0.3494
Epoch 30/100
450/460 [============================>.] - ETA: 0s - loss: 1.1017 - accuracy: 0.35
78
Epoch 30: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.1002 - accuracy:
0.3567 - val_loss: 548.4379 - val_accuracy: 0.3494
Epoch 31/100
454/460 [============================>.] - ETA: 0s - loss: 1.0958 - accuracy: 0.36
15
Epoch 31: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0971 - accuracy:
0.3600 - val_loss: 548.9719 - val_accuracy: 0.3494
Epoch 32/100
449/460 [============================>.] - ETA: 0s - loss: 1.0945 - accuracy: 0.41
87
Epoch 32: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0935 - accuracy:
0.4174 - val_loss: 549.5565 - val_accuracy: 0.3494
Epoch 33/100
452/460 [============================>.] - ETA: 0s - loss: 1.0942 - accuracy: 0.35
59
Epoch 33: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3567 - val_loss: 549.5576 - val_accuracy: 0.3494
Epoch 34/100
433/460 [===========================>..] - ETA: 0s - loss: 1.0929 - accuracy: 0.35
83
Epoch 34: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
```

```
0.3567 - val_loss: 549.5571 - val_accuracy: 0.3494
Epoch 35/100
436/460 [===========================>..] - ETA: 0s - loss: 1.0935 - accuracy: 0.37
33
Epoch 35: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3722 - val_loss: 549.5576 - val_accuracy: 0.3494
Epoch 36/100
456/460 [============================>.] - ETA: 0s - loss: 1.0934 - accuracy: 0.35
55
Epoch 36: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3565 - val_loss: 549.5576 - val_accuracy: 0.3494
Epoch 37/100
450/460 [============================>.] - ETA: 0s - loss: 1.0928 - accuracy: 0.37
83
Epoch 37: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0931 - accuracy:
0.3780 - val_loss: 549.5582 - val_accuracy: 0.3494
Epoch 38/100
438/460 [===========================>..] - ETA: 0s - loss: 1.0948 - accuracy: 0.34
19
Epoch 38: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3434 - val_loss: 549.5587 - val_accuracy: 0.3494
Epoch 39/100
451/460 [============================>.] - ETA: 0s - loss: 1.0940 - accuracy: 0.32
48
Epoch 39: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3238 - val_loss: 549.5576 - val_accuracy: 0.3494
Epoch 40/100
458/460 [============================>.] - ETA: 0s - loss: 1.0926 - accuracy: 0.34
63
Epoch 40: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3461 - val_loss: 549.5571 - val_accuracy: 0.3494
Epoch 41/100
440/460 [===========================>..] - ETA: 0s - loss: 1.0956 - accuracy: 0.36
14
Epoch 41: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3608 - val_loss: 549.5571 - val_accuracy: 0.3494
Epoch 42/100
436/460 [===========================>..] - ETA: 0s - loss: 1.0922 - accuracy: 0.42
14
Epoch 42: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0931 - accuracy:
0.4201 - val_loss: 549.5568 - val_accuracy: 0.3494
Epoch 43/100
438/460 [===========================>..] - ETA: 0s - loss: 1.0941 - accuracy: 0.39
81
Epoch 43: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3967 - val_loss: 549.5576 - val_accuracy: 0.3494
Epoch 44/100
```

```
457/460 [============================>.] - ETA: 0s - loss: 1.0932 - accuracy: 0.42
83
Epoch 44: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.4283 - val_loss: 549.5582 - val_accuracy: 0.3494
Epoch 45/100
433/460 [===========================>..] - ETA: 0s - loss: 1.0922 - accuracy: 0.35
91
Epoch 45: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3567 - val_loss: 549.5582 - val_accuracy: 0.3494
Epoch 46/100
444/460 [===========================>..] - ETA: 0s - loss: 1.0902 - accuracy: 0.36
04
Epoch 46: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3567 - val_loss: 549.5573 - val_accuracy: 0.3494
Epoch 47/100
442/460 [===========================>..] - ETA: 0s - loss: 1.0921 - accuracy: 0.39
88
Epoch 47: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3937 - val_loss: 549.5585 - val_accuracy: 0.3494
Epoch 48/100
447/460 [============================>.] - ETA: 0s - loss: 1.0931 - accuracy: 0.38
62
Epoch 48: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0931 - accuracy:
0.3872 - val_loss: 549.5586 - val_accuracy: 0.4376
Epoch 49/100
438/460 [===========================>..] - ETA: 0s - loss: 1.0930 - accuracy: 0.35
50
Epoch 49: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0931 - accuracy:
0.3578 - val_loss: 549.5571 - val_accuracy: 0.4376
Epoch 50/100
451/460 [============================>.] - ETA: 0s - loss: 1.0933 - accuracy: 0.32
76
Epoch 50: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3273 - val_loss: 549.5577 - val_accuracy: 0.3494
Epoch 51/100
454/460 [============================>.] - ETA: 0s - loss: 1.0942 - accuracy: 0.37
47
Epoch 51: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3750 - val_loss: 549.5573 - val_accuracy: 0.3494
Epoch 52/100
448/460 [============================>.] - ETA: 0s - loss: 1.0932 - accuracy: 0.39
76
Epoch 52: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3962 - val_loss: 549.5577 - val_accuracy: 0.3494
Epoch 53/100
446/460 [============================>.] - ETA: 0s - loss: 1.0946 - accuracy: 0.34
73
```

```
Epoch 53: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0931 - accuracy:
0.3459 - val_loss: 549.5575 - val_accuracy: 0.3494
Epoch 54/100
435/460 [===========================>..] - ETA: 0s - loss: 1.0938 - accuracy: 0.42
76
Epoch 54: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.4242 - val_loss: 549.5576 - val_accuracy: 0.3494
Epoch 55/100
458/460 [============================>.] - ETA: 0s - loss: 1.0927 - accuracy: 0.35
70
Epoch 55: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3565 - val_loss: 549.5568 - val_accuracy: 0.3494
Epoch 56/100
447/460 [============================>.] - ETA: 0s - loss: 1.0941 - accuracy: 0.36
55
Epoch 56: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3638 - val_loss: 549.5583 - val_accuracy: 0.3494
Epoch 57/100
451/460 [============================>.] - ETA: 0s - loss: 1.0938 - accuracy: 0.33
48
Epoch 57: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3358 - val_loss: 549.5576 - val_accuracy: 0.3494
Epoch 58/100
447/460 [============================>.] - ETA: 0s - loss: 1.0942 - accuracy: 0.34
79
Epoch 58: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3488 - val_loss: 549.5576 - val_accuracy: 0.3494
Epoch 59/100
451/460 [============================>.] - ETA: 0s - loss: 1.0930 - accuracy: 0.35
73
Epoch 59: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3567 - val_loss: 549.5569 - val_accuracy: 0.3494
Epoch 60/100
430/460 [===========================>..] - ETA: 0s - loss: 1.0926 - accuracy: 0.35
93
Epoch 60: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3576 - val_loss: 549.5565 - val_accuracy: 0.4376
Epoch 61/100
445/460 [============================>.] - ETA: 0s - loss: 1.0936 - accuracy: 0.40
81
Epoch 61: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.4063 - val_loss: 549.5576 - val_accuracy: 0.3494
Epoch 62/100
451/460 [============================>.] - ETA: 0s - loss: 1.0937 - accuracy: 0.42
46
Epoch 62: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0931 - accuracy:
```

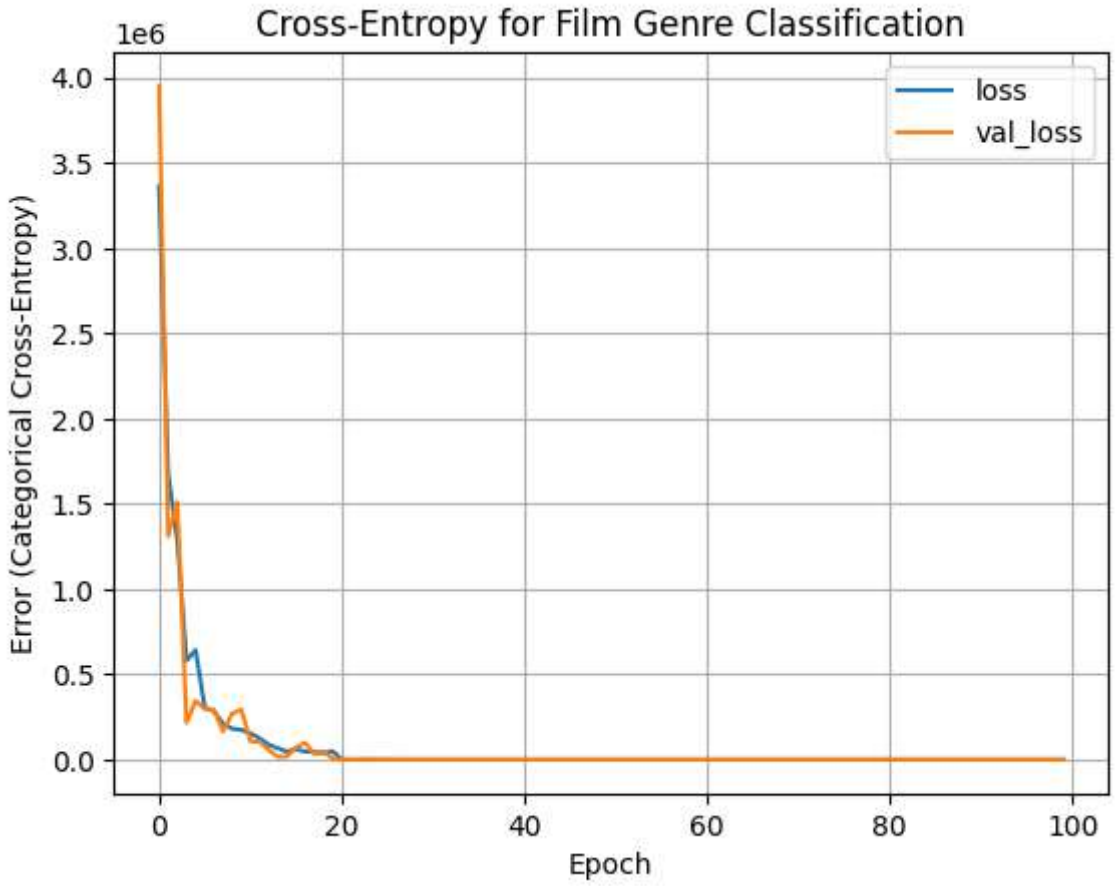
```
0.4231 - val_loss: 549.5580 - val_accuracy: 0.3494
Epoch 63/100
442/460 [===========================>..] - ETA: 0s - loss: 1.0933 - accuracy: 0.36
28
Epoch 63: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0931 - accuracy:
0.3600 - val_loss: 549.5579 - val_accuracy: 0.3494
Epoch 64/100
455/460 [============================>.] - ETA: 0s - loss: 1.0938 - accuracy: 0.42
69
Epoch 64: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.4267 - val_loss: 549.5580 - val_accuracy: 0.3494
Epoch 65/100
436/460 [===========================>..] - ETA: 0s - loss: 1.0926 - accuracy: 0.38
85
Epoch 65: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3886 - val_loss: 549.5582 - val_accuracy: 0.3494
Epoch 66/100
437/460 [===========================>..] - ETA: 0s - loss: 1.0923 - accuracy: 0.35
70
Epoch 66: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3567 - val_loss: 549.5583 - val_accuracy: 0.3494
Epoch 67/100
456/460 [============================>.] - ETA: 0s - loss: 1.0931 - accuracy: 0.37
23
Epoch 67: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3722 - val_loss: 549.5582 - val_accuracy: 0.3494
Epoch 68/100
459/460 [============================>.] - ETA: 0s - loss: 1.0932 - accuracy: 0.39
22
Epoch 68: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3921 - val_loss: 549.5582 - val_accuracy: 0.3494
Epoch 69/100
436/460 [===========================>..] - ETA: 0s - loss: 1.0923 - accuracy: 0.35
58
Epoch 69: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3548 - val_loss: 549.5583 - val_accuracy: 0.3494
Epoch 70/100
445/460 [============================>.] - ETA: 0s - loss: 1.0943 - accuracy: 0.35
93
Epoch 70: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3581 - val_loss: 549.5590 - val_accuracy: 0.3494
Epoch 71/100
437/460 [===========================>..] - ETA: 0s - loss: 1.0936 - accuracy: 0.38
44
Epoch 71: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0931 - accuracy:
0.3807 - val_loss: 549.5595 - val_accuracy: 0.3494
Epoch 72/100
```

```
458/460 [============================>.] - ETA: 0s - loss: 1.0937 - accuracy: 0.42
63
Epoch 72: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0931 - accuracy:
0.4259 - val_loss: 549.5617 - val_accuracy: 0.3494
Epoch 73/100
451/460 [============================>.] - ETA: 0s - loss: 1.0942 - accuracy: 0.36
00
Epoch 73: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3597 - val_loss: 549.5624 - val_accuracy: 0.3494
Epoch 74/100
433/460 [============================>..] - ETA: 0s - loss: 1.0947 - accuracy: 0.35
80
Epoch 74: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3567 - val_loss: 549.5626 - val_accuracy: 0.3494
Epoch 75/100
436/460 [============================>..] - ETA: 0s - loss: 1.0930 - accuracy: 0.40
40
Epoch 75: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3997 - val_loss: 549.5634 - val_accuracy: 0.3494
Epoch 76/100
439/460 [============================>..] - ETA: 0s - loss: 1.0941 - accuracy: 0.37
19
Epoch 76: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3709 - val_loss: 549.5660 - val_accuracy: 0.3494
Epoch 77/100
458/460 [============================>.] - ETA: 0s - loss: 1.0928 - accuracy: 0.34
33
Epoch 77: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3434 - val_loss: 549.5671 - val_accuracy: 0.3494
Epoch 78/100
460/460 [==============================] - ETA: 0s - loss: 1.0930 - accuracy: 0.32
19
Epoch 78: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3219 - val_loss: 549.5699 - val_accuracy: 0.3494
Epoch 79/100
444/460 [============================>..] - ETA: 0s - loss: 1.0942 - accuracy: 0.40
93
Epoch 79: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.4065 - val_loss: 549.5751 - val_accuracy: 0.3494
Epoch 80/100
440/460 [============================>..] - ETA: 0s - loss: 1.0947 - accuracy: 0.38
78
Epoch 80: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3850 - val_loss: 549.5736 - val_accuracy: 0.3494
Epoch 81/100
438/460 [============================>..] - ETA: 0s - loss: 1.0938 - accuracy: 0.38
47
```

```
Epoch 81: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3839 - val_loss: 549.5807 - val_accuracy: 0.3494
Epoch 82/100
444/460 [===========================>..] - ETA: 0s - loss: 1.0921 - accuracy: 0.35
90
Epoch 82: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3567 - val_loss: 549.5772 - val_accuracy: 0.3494
Epoch 83/100
442/460 [===========================>..] - ETA: 0s - loss: 1.0929 - accuracy: 0.37
53
Epoch 83: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3744 - val_loss: 549.5745 - val_accuracy: 0.3494
Epoch 84/100
441/460 [===========================>..] - ETA: 0s - loss: 1.0939 - accuracy: 0.34
13
Epoch 84: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3423 - val_loss: 549.5891 - val_accuracy: 0.3494
Epoch 85/100
450/460 [============================>.] - ETA: 0s - loss: 1.0927 - accuracy: 0.37
50
Epoch 85: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3758 - val_loss: 549.5854 - val_accuracy: 0.4376
Epoch 86/100
439/460 [===========================>..] - ETA: 0s - loss: 1.0942 - accuracy: 0.32
49
Epoch 86: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3271 - val_loss: 549.5789 - val_accuracy: 0.4376
Epoch 87/100
450/460 [============================>.] - ETA: 0s - loss: 1.0941 - accuracy: 0.42
03
Epoch 87: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.4190 - val_loss: 549.5686 - val_accuracy: 0.3494
Epoch 88/100
442/460 [===========================>..] - ETA: 0s - loss: 1.0935 - accuracy: 0.35
92
Epoch 88: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0931 - accuracy:
0.3608 - val_loss: 549.6040 - val_accuracy: 0.4376
Epoch 89/100
449/460 [============================>.] - ETA: 0s - loss: 1.0931 - accuracy: 0.34
86
Epoch 89: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3483 - val_loss: 549.5964 - val_accuracy: 0.3494
Epoch 90/100
452/460 [============================>.] - ETA: 0s - loss: 1.0940 - accuracy: 0.35
67
Epoch 90: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0931 - accuracy:
```

```
0.3556 - val_loss: 549.5882 - val_accuracy: 0.3494
Epoch 91/100
455/460 [============================>.] - ETA: 0s - loss: 1.0931 - accuracy: 0.37
83
Epoch 91: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0931 - accuracy:
0.3785 - val_loss: 549.5705 - val_accuracy: 0.3494
Epoch 92/100
446/460 [============================>.] - ETA: 0s - loss: 1.0924 - accuracy: 0.32
68
Epoch 92: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3263 - val_loss: 549.5347 - val_accuracy: 0.3494
Epoch 93/100
449/460 [============================>.] - ETA: 0s - loss: 1.0947 - accuracy: 0.36
78
Epoch 93: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3698 - val_loss: 549.6776 - val_accuracy: 0.4376
Epoch 94/100
445/460 [============================>.] - ETA: 0s - loss: 1.0919 - accuracy: 0.35
28
Epoch 94: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3524 - val_loss: 549.6770 - val_accuracy: 0.3494
Epoch 95/100
443/460 [============================>..] - ETA: 0s - loss: 1.0911 - accuracy: 0.37
22
Epoch 95: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3712 - val_loss: 549.6766 - val_accuracy: 0.3494
Epoch 96/100
443/460 [============================>..] - ETA: 0s - loss: 1.0952 - accuracy: 0.30
36
Epoch 96: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3059 - val_loss: 549.6758 - val_accuracy: 0.3494
Epoch 97/100
455/460 [============================>.] - ETA: 0s - loss: 1.0936 - accuracy: 0.35
52
Epoch 97: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3548 - val_loss: 549.6752 - val_accuracy: 0.3494
Epoch 98/100
448/460 [============================>.] - ETA: 0s - loss: 1.0918 - accuracy: 0.36
52
Epoch 98: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0931 - accuracy:
0.3638 - val_loss: 549.6744 - val_accuracy: 0.3494
Epoch 99/100
449/460 [============================>.] - ETA: 0s - loss: 1.0923 - accuracy: 0.37
39
Epoch 99: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3717 - val_loss: 549.6743 - val_accuracy: 0.3494
Epoch 100/100
```

```
439/460 [===========================>..] - ETA: 0s - loss: 1.0916 - accuracy: 0.39
72
Epoch 100: saving model to training\cp.ckpt
460/460 [==============================] - 1s 2ms/step - loss: 1.0930 - accuracy:
0.3959 - val_loss: 549.6738 - val_accuracy: 0.3494
```



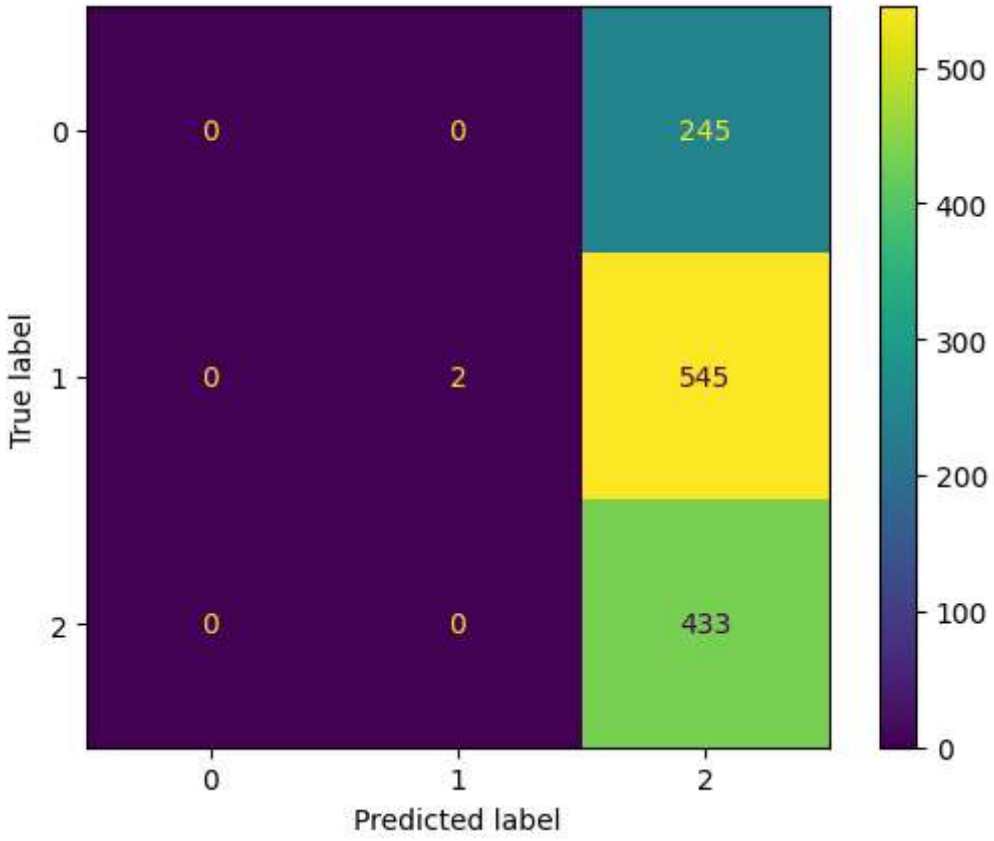

```
39/39 [==============================] - 0s 1ms/step - loss: 13.6252 - accuracy:
0.3551
[13.625152587890625, 0.3551020324230194]
```

Great, now that we've trained our model, we can evaluate it on our test set.

In [ ]:
```
pred_model = keras.Sequential([model, layers.Softmax()])

test_pred = pred_model.predict(test_x)

# Randomly sample 10 predicted softmax outputs
index = np.random.choice(test_pred.shape[0], 10, replace=False)

display(test_pred[index])

test_pred_vals = np.argmax(test_pred, axis=1)

cn_matrix = confusion_matrix(test_y, test_pred_vals)
disp = ConfusionMatrixDisplay(cn_matrix)
disp.plot()
plt.savefig('./imgs/nn_ims/confusion_matrix.png')
plt.show()
```

```
print(f"Test accuracy: {accuracy_score(test_y, test_pred_vals)}")
```

```
39/39 [==============================] - 0s 1ms/step
array([[0.33179682, 0.3340593 , 0.33414382],
       [0.33179682, 0.3340593 , 0.33414382],
       [0.33179682, 0.3340593 , 0.33414382],
       [0.33179682, 0.3340593 , 0.33414382],
       [0.33179682, 0.3340593 , 0.33414382],
       [0.33179682, 0.3340593 , 0.33414382],
       [0.33179682, 0.3340593 , 0.33414382],
       [0.33179682, 0.3340593 , 0.33414382],
       [0.33179682, 0.3340593 , 0.33414382],
       [0.33179682, 0.3340593 , 0.33414382]], dtype=float32)
```




```
Test accuracy: 0.3551020408163265
```

Notice above that we don't do much better than just predicting randomly. This is interesting, and suggests that we may just be dealing with data that can't be predicted on. After **various** tweaks and attempts to get better performance, the model always tended toward this type of distribution, suggesting that the ideal loss minimization is to slightly prioritize one class over another.