Universidad de los Andes

MIIA4406 - MOVIE GENRE CLASSIFICATION

MOVIE GENRE CLASSIFICATION

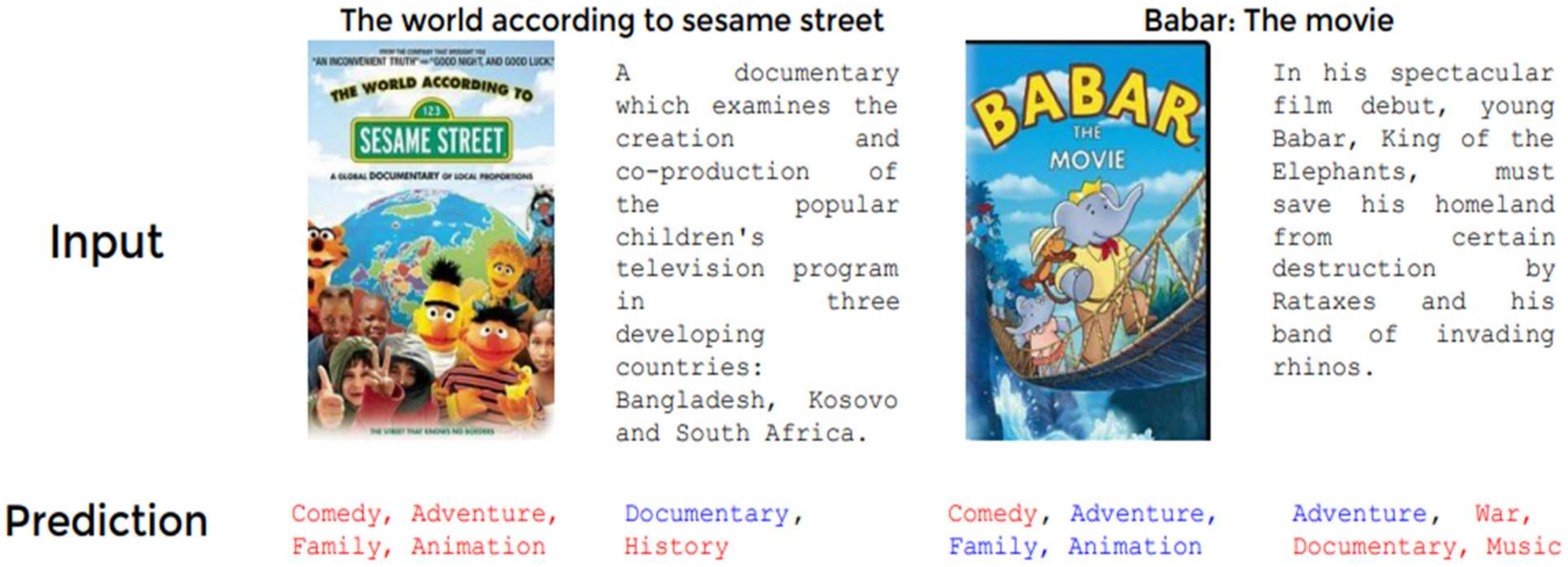
Camilo Prada Ladino cod. 201021153

Erika Marcela Ángel cod. 201628012

About the Competition

Movie Genre Classification

Classify a movie genre based on its plot and its poster.



	The world according to sesame street		Babar: The movie	
Input		A documentary which examines the creation and co-production of the popular children's television program in three developing countries: Bangladesh, Kosovo and South Africa.		In his spectacular film debut, young Babar, King of the Elephants, must save his homeland from certain destruction by Rataxes and his band of invading rhinos.
Prediction	Comedy, Adventure, Family, Animation	Documentary, History	Comedy, Adventure, Family, Animation	Adventure, War, Documentary, Music

Evaluation

MCAUC: Mean Columnwise Area Under Receiver Operating Characteristic

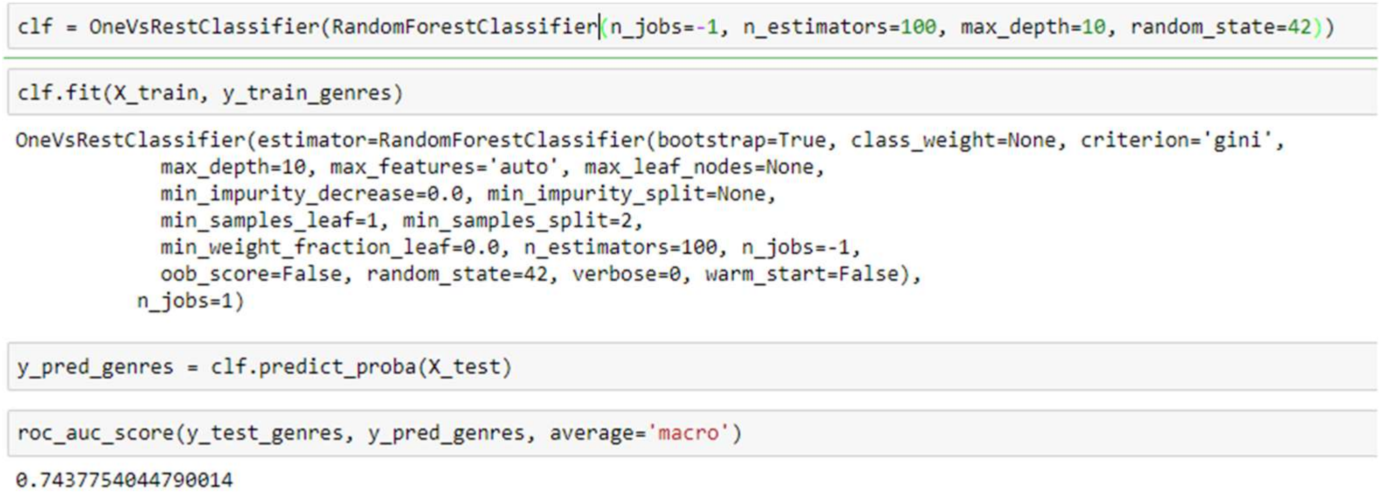
in sklearn:

roc_auc_score(y_test_genres, y_pred_genres, average='macro')

Universidad de los Andes

Img Text Machine learning

Ramdom Forest (Example)



```
clf = OneVsRestClassifier(RandomForestClassifier(n_jobs=-1, n_estimators=100, max_depth=10, random_state=42))
```

```
clf.fit(X_train, y_train_genres)
```

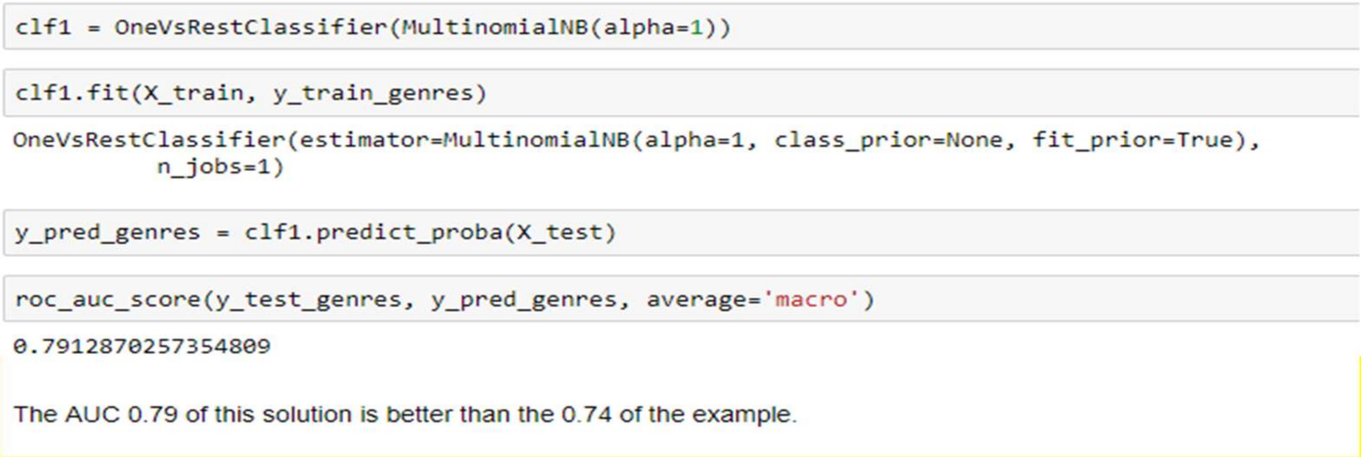
```
OneVsRestClassifier(estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1,
            oob_score=False, random_state=42, verbose=0, warm_start=False),
          n_jobs=1)
```

```
y_pred_genres = clf.predict_proba(X_test)
```

```
roc_auc_score(y_test_genres, y_pred_genres, average='macro')
```

```
0.7437754044790014
```

Multinomial NB



```
clf1 = OneVsRestClassifier(MultinomialNB(alpha=1))
```

```
clf1.fit(X_train, y_train_genres)
```

```
OneVsRestClassifier(estimator=MultinomialNB(alpha=1, class_prior=None, fit_prior=True),
          n_jobs=1)
```

```
y_pred_genres = clf1.predict_proba(X_test)
```

```
roc_auc_score(y_test_genres, y_pred_genres, average='macro')
```

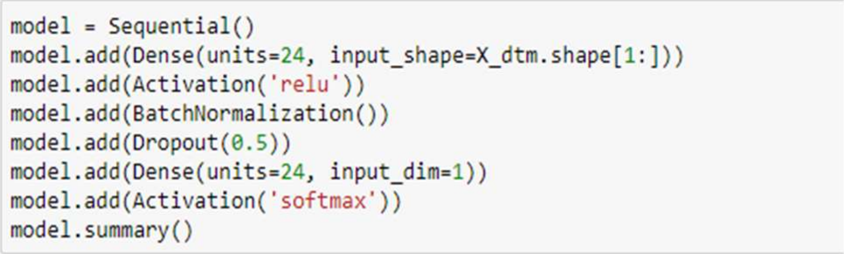
```
0.7912870257354809
```

The AUC 0.79 of this solution is better than the 0.74 of the example.

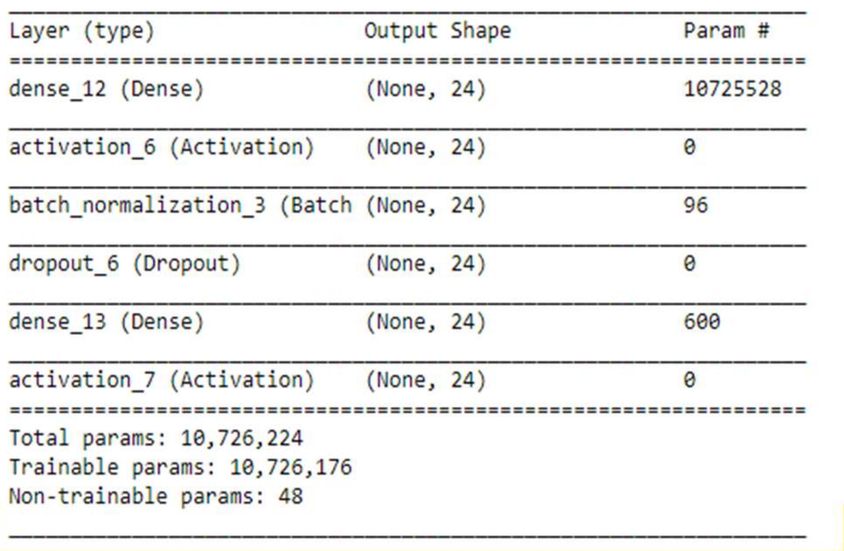
Universidad de los Andes

Img Text CNN

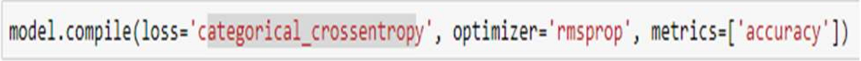
The Network
Now than we already have the Y and X´s for our algorithm we configurate a NN with the imputs of the vectorized words and the units gonna be 24 for the categories of the "genres" in the training, we build two levels for this network the first has a "relu" activation function, and the second level have a "softmax" activation function.



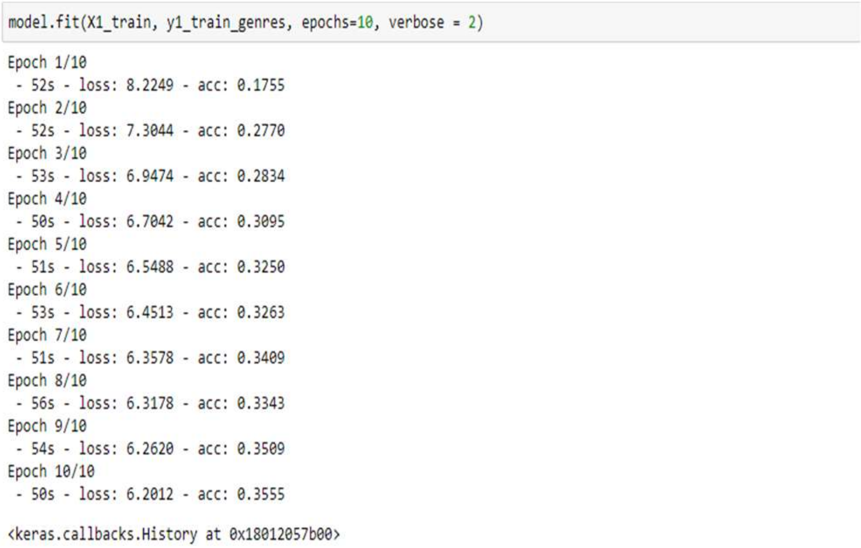
```
model = Sequential()
model.add(Dense(units=24, input_shape=X_dtm.shape[1:]))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(units=24, input_dim=1))
model.add(Activation('softmax'))
model.summary()
```



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_12 (Dense)             (None, 24)                10725528
_________________________________________________________________
activation_6 (Activation)    (None, 24)                0
_________________________________________________________________
batch_normalization_3 (Batch (None, 24)                96
_________________________________________________________________
dropout_6 (Dropout)          (None, 24)                0
_________________________________________________________________
dense_13 (Dense)             (None, 24)                600
_________________________________________________________________
activation_7 (Activation)    (None, 24)                0
=================================================================
Total params: 10,726,224
Trainable params: 10,726,176
Non-trainable params: 48
_________________________________________________________________
```



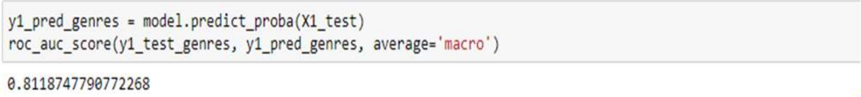
```
model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
```



```
model.fit(X1_train, y1_train_genres, epochs=10, verbose = 2)
```

```
Epoch 1/10
 - 52s - loss: 8.2249 - acc: 0.1755
Epoch 2/10
 - 52s - loss: 7.3044 - acc: 0.2770
Epoch 3/10
 - 53s - loss: 6.9474 - acc: 0.2834
Epoch 4/10
 - 50s - loss: 6.7042 - acc: 0.3095
Epoch 5/10
 - 51s - loss: 6.5488 - acc: 0.3250
Epoch 6/10
 - 53s - loss: 6.4513 - acc: 0.3263
Epoch 7/10
 - 51s - loss: 6.3578 - acc: 0.3409
Epoch 8/10
 - 56s - loss: 6.3178 - acc: 0.3343
Epoch 9/10
 - 54s - loss: 6.2620 - acc: 0.3509
Epoch 10/10
 - 50s - loss: 6.2012 - acc: 0.3555

<keras.callbacks.History at 0x18012057b00>
```



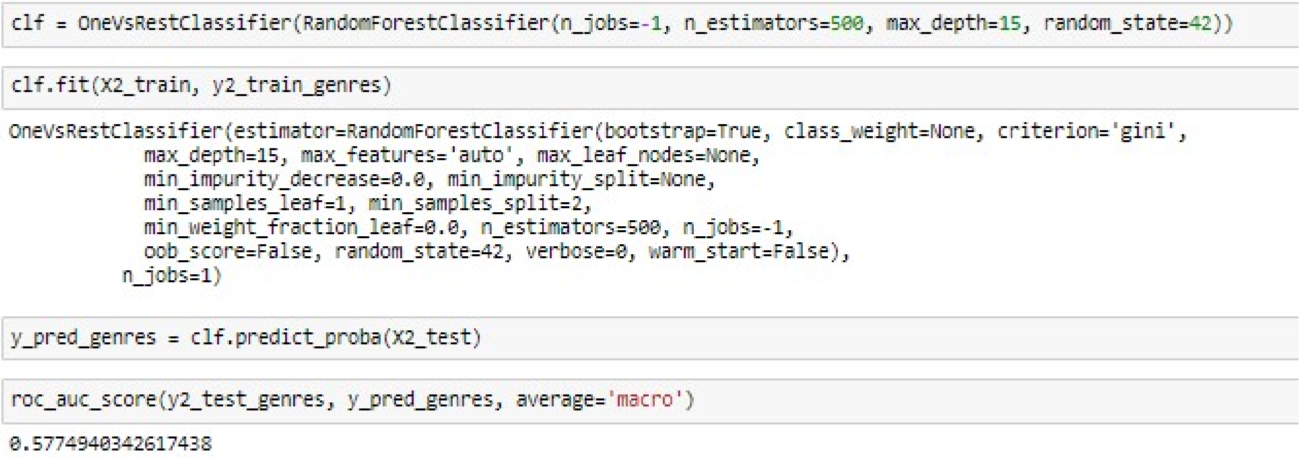
```
y1_pred_genres = model.predict_proba(X1_test)
roc_auc_score(y1_test_genres, y1_pred_genres, average='macro')
```

```
0.8118747790772268
```

Finally once we evaluate the network, we find a AUC of 0.81. In kaggle the score was 0.80544 ang was the best score for the group.

Img Machine learning

Ramdom Forest



```
clf = OneVsRestClassifier(RandomForestClassifier(n_jobs=-1, n_estimators=500, max_depth=15, random_state=42))
```

```
clf.fit(X2_train, y2_train_genres)
```

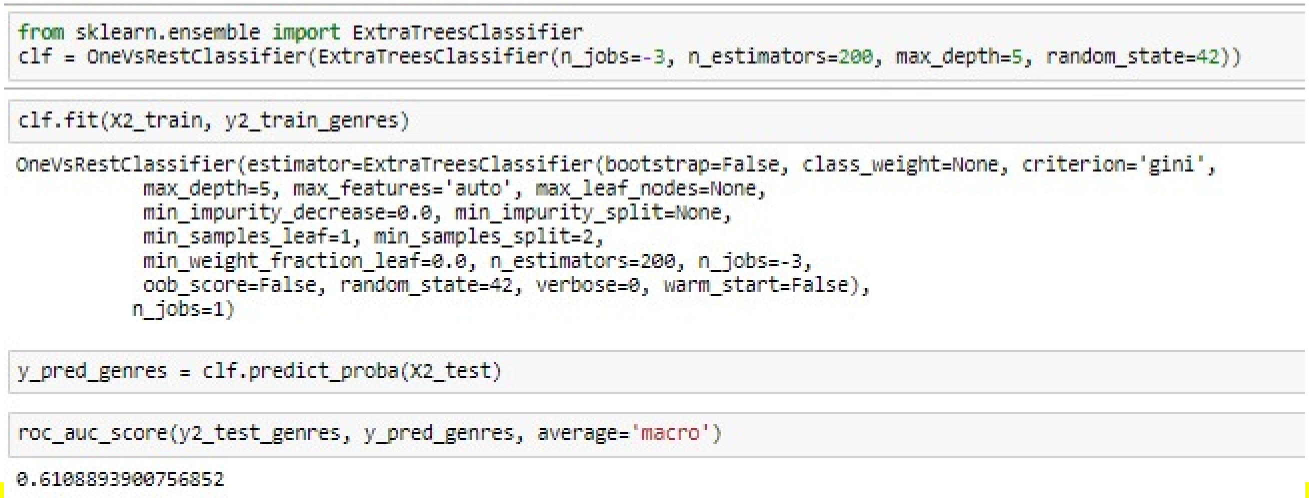
```
OneVsRestClassifier(estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=15, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=-1,
            oob_score=False, random_state=42, verbose=0, warm_start=False),
          n_jobs=1)
```

```
y_pred_genres = clf.predict_proba(X2_test)
```

```
roc_auc_score(y2_test_genres, y_pred_genres, average='macro')
```

```
0.5774940342617438
```

Extra Trees Clasifier



```
from sklearn.ensemble import ExtraTreesClassifier
clf = OneVsRestClassifier(ExtraTreesClassifier(n_jobs=-3, n_estimators=200, max_depth=5, random_state=42))
```

```
clf.fit(X2_train, y2_train_genres)
```

```
OneVsRestClassifier(estimator=ExtraTreesClassifier(bootstrap=False, class_weight=None, criterion='gini',
           max_depth=5, max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=200, n_jobs=-3,
           oob_score=False, random_state=42, verbose=0, warm_start=False),
          n_jobs=1)
```

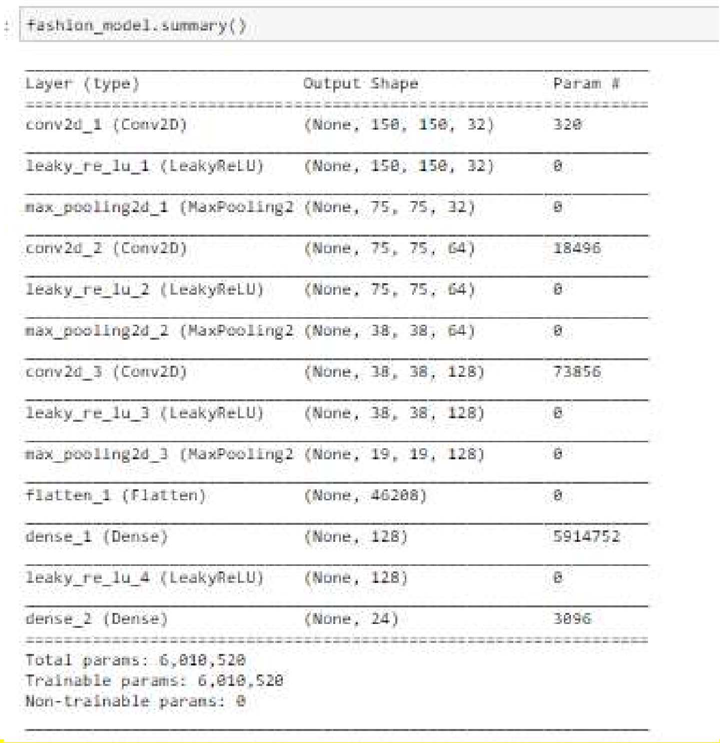
```
y_pred_genres = clf.predict_proba(X2_test)
```

```
roc_auc_score(y2_test_genres, y_pred_genres, average='macro')
```

```
0.6108893900756852
```

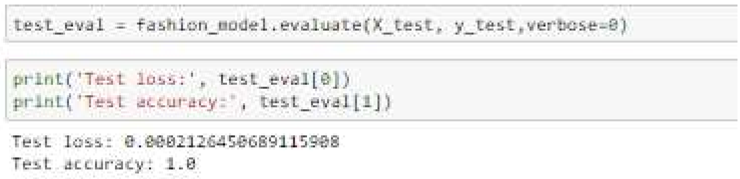
Img CNN

The Network
The images are of size 150 x 150. we convert the image matrix to an array, rescale it between 0 and 1, reshape it so that it's of size 150 x 150 x 1, and feed this as an input to the network.
We use three convolutional layers:
The first layer will have 32-3 x 3 filters, The second layer will have 64-3 x 3 filters and The third layer will have 128-3 x 3 filters. In addition, there are three max-pooling layers each of size 2 x 2.



```
fashion_model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 150, 150, 32)      320
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 150, 150, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 75, 75, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 75, 75, 64)        18496
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 75, 75, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 38, 38, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 38, 38, 128)       73856
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 38, 38, 128)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 19, 19, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 46208)             0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               5914752
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 24)                3096
=================================================================
Total params: 6,010,520
Trainable params: 6,010,520
Non-trainable params: 0
_________________________________________________________________
```



```
test_eval = fashion_model.evaluate(X_test, y_test,verbose=0)
```

```
print('Test loss:', test_eval[0])
print('Test accuracy:', test_eval[1])
```

```
Test loss: 0.00021264506089115908
Test accuracy: 1.0
```

the test accuracy looks good. However, the model looked like it was overfitting, so for that problem we added Dropout into the Network but it did not work.