國立臺灣科技大學資訊工程系

109 學年度第二學期專題研究

總報告

Movie Genre Classification

研究組員

B10715008 黃少聰

B10715009 孫麗珠

指導教授： Kenneth Pao (鮑興國)

中 華 民 國 110 年 06 月 26 日

1 INTRODUCTION

Movie genre classification is an interesting problem to solve, and with a large number of movies in the market, it is tedious to mark the genres of movies by watching each movie one by one. Therefore, in this project, we made a movie genre classification program that classified movies based on a movie's poster, subtitle, and trailer.

2 METHODS

2.1 Movie Posters

Movie posters are a key component in the film industry. It is a primary design that captures the viewer's attention and conveys a movie. Human emotions are aroused by colour, brightness, saturation, hues, contours, etc in images. Therefore, we are able to quickly draw about a movie's genre (comedy, action, drama, animation, etc) based on the colours, facial expressions, and scenes portrayed on a movie poster. This leads to the assumption that the colour information, texture features, and structural cues contained in images of posters, possess some inherent relationship that could be exploited by Machine Learning algorithms for automated prediction of movie genre from posters.

This is multi-label classification task, since a movie can have multiple genres linked to it, i.e., have an independent probability to belong to each label (genre). The class labels (i.e., the genres) are categorical in nature and have to be converted to numerical form before classification is performed. Multi-hot encoding is adopted, which converts categorical labels into a vector of binary values. 9 unique genres are found and each genre is represented as a one-hot encoded column. If a movie belongs to a genre, the value is 1, else 0. As an image can belong to multiple genres, here it a case of multiple-hot encoding (as multiple genre values can be "hot"). After transformation, the encoded labels look like this:

	movie	genre	disaster	scifi	adventure	comedy	action	horror	romance	spy	martialarts
0	100earthqua	['disaster']	1	0	0	0	0	0	0	0	0
1	2012	['disaster', 's	1	1	0	0	0	0	0	0	0
2	advcom-rio	['adventure',	0	0	1	1	0	0	0	0	0
3	adventuresin	['adventure']	0	0	1	0	0	0	0	0	0
4	alitabattlean	['scifi', 'actio	0	1	0	0	1	0	0	0	0
5	annihilation	['scifi', 'horro	0	1	0	0	0	1	0	0	0
6	awalktoreme	['romance']	0	0	0	0	0	0	1	0	0
7	billandtedfac	['scifi', 'come	0	1	0	1	0	0	0	0	0
8	casinoroyale	['action', 'spy	0	0	0	0	1	0	0	1	0
9	centralintelli	['comedy', 'a	0	0	0	1	1	0	0	0	0
10	coldskin	['scifi', 'horro	0	1	0	0	0	1	0	0	0
11	crazyrichasia	['comedy', 'r	0	0	0	1	0	0	1	0	0
12	dayofthedea	['action', 'ho	0	0	0	0	1	1	0	0	0
13	deepbluesea	['scifi', 'horro	0	1	0	0	0	1	0	0	0

In this project, fine-tuning pre-trained VGG16 is proposed. VGG16 is loaded with pre-trained weights (imagenet) and without the classifier layers (top layer). All the layers, except the last 4, are then frozen. Finally, to this VGG convolutional model, an fully connected classifier layer is added followed by a sigmoid layer with 9 outputs. For an input image shape of (200,150,3), the model summary is as follows:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 6, 4, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 12288)             0
_________________________________________________________________
dense (Dense)                (None, 1024)              12583936
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 9)                 9225
=================================================================
Total params: 27,307,849
Trainable params: 17,312,777
Non-trainable params: 9,995,072
```

An ImageDataGenerator is prepared before training, to perform data augmentation. The model is trained for 30 epochs, using RMSProp as optimizer (with 1e-5 learning rate) and binary cross entropy as loss.

```
model.compile(optimizer=optimizers.RMSprop(learning_rate=1e-5), loss='binary_crossentropy', metrics=['accuracy'])
```

```
aug = ImageDataGenerator(rotation_range=20, zoom_range=0.15,width_shift_range=0.2, height_shift_range=0.2,
                         shear_range=0.15,horizontal_flip=True, fill_mode="nearest")
```

```
import tensorflow as tf
```

```
EPOCHS=30
BS = 4

history = model.fit(aug.flow(X_train, Y_train, batch_size=BS),validation_data=(X_valid, Y_valid),
                    steps_per_epoch=len(X_train) // BS, epochs=EPOCHS)

model.save('poster-model.h5')
```

2.2 Movie Subtitle

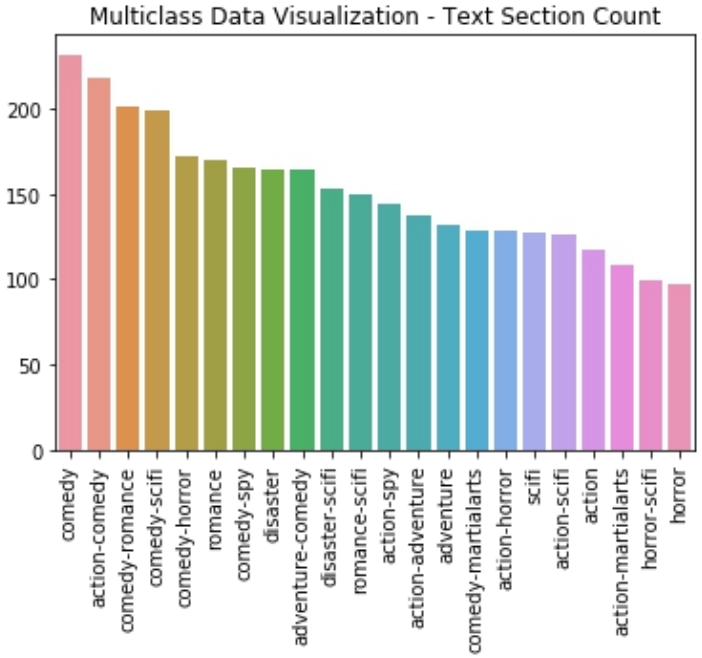
Movie subtitles are also important and is often key in recognizing a movie's genre, since they contain the entire dialogues and even descriptions on the movie's audio (i.e. in subtitles for the hearing impaired). Natural Language Processing (NLP) is hence used to classify movie subtitles into different genres.

For this multiclass task, a BERT-based model is proposed. Since the input subtitle files (.srt) can be quite long, it cannot entirely fit within BERT's maximum sequence length of 512 tokens. To work around this, we separated the input text into sections of about 200 words each, with an overlap of 50 words between one section and the next.

Each of the sections are then encoded using HuggingFace Transformers' BertTokenizer. Pytorch BucketIterators and TabularDatasets are used to prepare the data from there on, and finally, HuggingFace Transformers' BertForSequenceClassification is used along with the pretrained weights from bert-base-uncased. We fine-tuned the model using our dataset of 110 subtitle files, each of which was divided into sections as described.

	movie	genre	text
0	100earthquake	['disaster']	Come on, Hicks. This is for your all-time best...
1	100earthquake	['disaster']	yours? I don't know which way you're facing! I...
2	100earthquake	['disaster']	and I would appreciate not being cut out of it...
3	100earthquake	['disaster']	haven't seen last night's seismographs. I have...
4	100earthquake	['disaster']	have firearms? -No. -Of course not! Fireworks?...
...	...	...	...
3328	voyagers	['scifi']	it the alien? It was Christopher. (BREATHING H...
3329	voyagers	['scifi']	GRUNTING) (ALL CLAMORING) Grab him. Get off me...
3330	voyagers	['scifi']	Shut up. CREW MEMBERS: Shut up. Shut up. Shut ...
3331	voyagers	['scifi']	released. (DRAMATIC MUSIC PLAYING) (SELA AND Z...
3332	voyagers	['scifi']	PLAYING) CREW MEMBER: Hey, Chief, we just star...

3333 rows × 3 columns



2.3 Movie Trailer (Frames)

For this task, the frames of each trailer is sampled and grouped as shown below:

One trailer could have anywhere from 60 to 80 of these frame grids.

The model architecture we used to classify the frame grids are as follows:

```
        self.conv1 = conv_block(in_channels, 64)
        self.conv2 = conv_block(64, 128, pool=True)
        self.res1 = nn.Sequential(
            conv_block(128, 128),
            conv_block(128, 128)
        )

        self.conv3 = conv_block(128, 256, pool=True)
        self.conv4 = conv_block(256, 512, pool=True)
        self.res2 = nn.Sequential(
            conv_block(512, 512),
            conv_block(512, 512)
        )

        self.classifier1 = nn.Sequential(
            nn.MaxPool2d(4),
            nn.Flatten()
        )

        self.classifier2 = nn.Sequential(
            nn.Dropout(0.5),
            nn.Linear(25088, 2048),
            nn.Dropout(0.25),
            nn.Linear(2048, num_classes)
        )

    def forward(self, xb):
        out = self.conv1(xb)
        out = self.conv2(out)
        out = self.res1(out) + out

        out = self.conv3(out)
        out = self.conv4(out)
        out = self.res2(out) + out

        out = self.classifier1(out) # output after flattened 4x25088
        out = self.classifier2(out)
```

```
def conv_block(in_channels, out_channels, pool=False):
    layers = [
                nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1),
                nn.BatchNorm2d(out_channels),
                nn.ReLU(inplace=True)
    ]
    if pool:
        layers.append(nn.MaxPool2d(2))
    return nn.Sequential(*layers)
```

Data augmentation is also done to help the model better generalize, which include random padding and cropping, and horizontal flipping with a probability of 50%. We trained the model for 10-20 epochs, with an 1e-4 learning rate, 0.1 gradient clip, and 1e-3 weight decay regularization. The optimizer we used is the AdamW optimizer from PyTorch. In addition to the above mentioned, the model was also trained using various schedulers such as the OneCycleLR and

ReduceLROnPlateau from PyTorch. Lastly, the loss function we used is the cross entropy loss from PyTorch.

2.4 Text Statistics & Facial Expressions

It is not only from the NLP aspect can we get features for movie genre classification. Hence, we also took measure of each subtitle's total sentence count, average sentence length, and average dialogue speed.

We combined this data with statistics on detected facial expressions obtained from the movie's trailers. The emotions counted are angry, disgusted, fear, happy, neutral, sad, and surprised, with an additional 'not-detected' emotion that could signify the lack of faces/people in a movie.

First, sentences were extracted by first combining the texts of a whole subtitle file into one paragraph and splitting whenever a period (.), an exclamation mark (!), or a question mark(?) is found. From there, the total sentence count, average sentence length, and average dialogue speed can be obtained using trivial algorithms.

Emotions are obtained by using a previous emotion detection model and its corresponding pretrained weights. From each trailer, a frame is taken every 0.5 seconds, which is then scanned for possible facial emotions.

total_sent_n	avg_dialogue	avg_word_p	ANG	DIS	FEA	HAP	ND	NEU	SAD	SUR
1610	2.94735402	26.0670808	3	0	8	7	134	8	19	0
1963	2.71789	29.7045339	1	0	3	3	320	3	15	0
1392	2.71372895	24.4748563	0	0	1	6	322	1	0	3
1087	2.72339725	24.9521619	0	0	0	1	207	0	0	0
1114	1.99031669	23.2962298	4	0	12	7	173	11	7	1
1706	2.40816128	23.2924971	12	0	16	5	224	10	22	0
1281	2.59632713	25.6604216	12	0	23	32	140	36	36	4
2956	3.38699079	24.4624493	15	0	50	66	152	7	22	6
584	2.20200566	25.2739726	9	0	28	16	177	19	49	1
1955	2.69922969	28.4644501	16	0	35	40	183	25	12	4
1079	2.45293098	24.1306766	6	0	9	6	261	18	30	0
1303	2.5385338	26	6	0	21	65	114	45	37	1
1532	2.37333679	24.5039165	6	0	4	1	242	2	21	1
1686	2.80885413	31.4934757	3	0	4	7	254	0	11	0
808	2.4785296	39.3279703	5	1	30	15	197	14	25	6
1592	2.15390773	24.6765075	16	0	35	30	162	12	44	4
806	2.38370627	32.9528536	8	0	7	3	144	7	8	3
943	2.72751859	19.5185578	0	0	1	4	256	5	2	7

The multi-hot encoded labels are then as follows:

disaster	scifi	action	horror	romance	comedy	spy	martialarts	adventure
1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	1	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0
0	0	1	0	0	1	0	0	0
0	1	0	1	0	0	0	0	0
0	0	0	0	1	1	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0
0	0	0	0	0	1	0	1	0

2.5 Combining

We combined the results of our various tasks by using a weighted average. Models with better prediction will be given a larger weight than models with less accuracy.

3 RESULTS

After extensive research and experimentations on various models and hyperparameters, we present below the results of our project.

3.1 Movie Posters

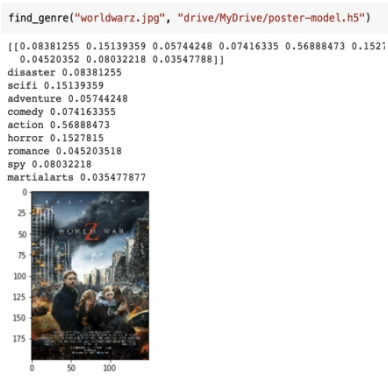
The model used for classifying movie posters achieved an accuracy of about 86.36%. Out of the 22 movies set aside for testing purposes, it can correctly identify the genres of 19.

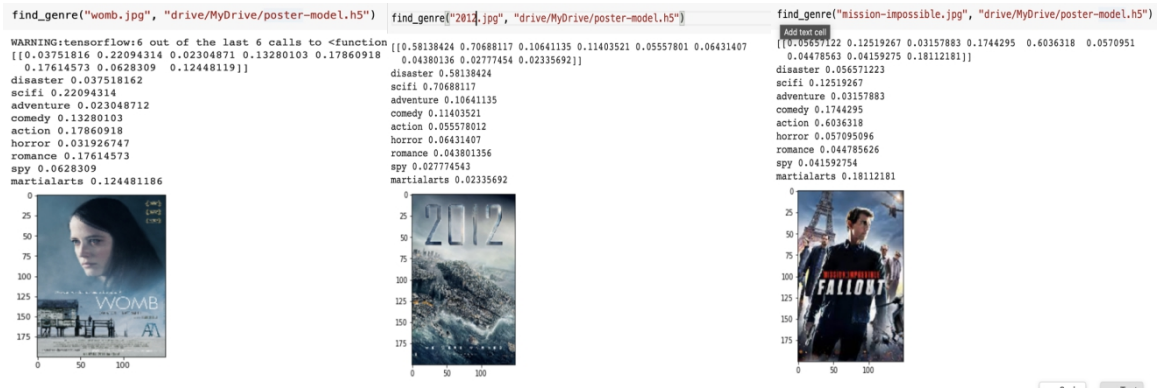
```
accuracy_score('drive/MyDrive/poster-dataset/finegrained_poster_test_data_multihotencoded.csv', 'drive/MyDrive/poster-dataset/poster-model.h5')
```

```
  0%|          | 0/22 [00:00<?, ?it/s]['jumanjiwelcometothejungle' 'paul' 'shaolinsoccer' 'spectre' 'sputnik'
 'thecore' 'thediscovery' 'theedgeofseventeen'
 'thefastandthefurioustokyodrift' 'thefinalmaster' 'thekissingbooth'
 'theperfectstorm' 'thephotograph' 'thering'
 'theseventhadventuresofsinbad' 'thismeanswar' 'triplefrontier'
 'tronlegacy' 'vampiresvsthebronx' 'womb' 'worldwarz' 'zoolander2']
100%|██████████| 22/22 [00:07<00:00,  2.79it/s]
Shape of images: (22, 200, 150, 3)
Shape of labels: (22, 9)
100%|██████████| 22/22 [00:00<00:00, 5974.79it/s]Images having atleast one genre correctly identified 19
Total number of images = 22
Accuracy =  0.8636363636363636
```

The following are test results, obtained by choosing the maximum two prediction scores, on single images:

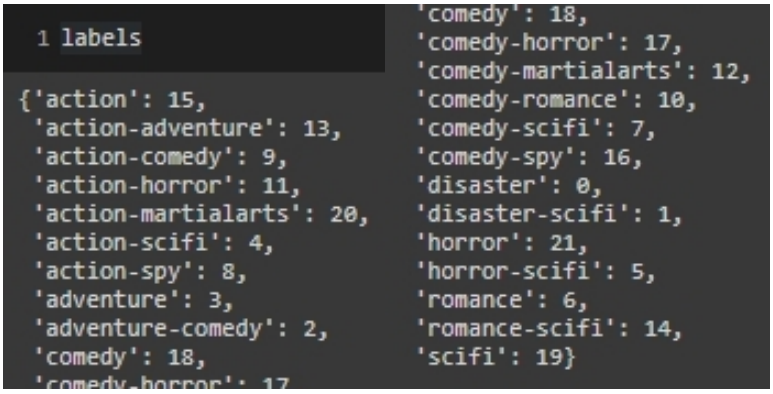
Movie Name	Actual Genre	Predicted Genre #1	Output Score for Genre #1	Predicted Genre #2	Output Score for Genre #2
The Ring	Horror	Horror	0.710849	Action	0.302006
World War Z	Action, Horror	Action	0.568884	Horror	0.152781
When in Rome	Romance, Comedy	Comedy	0.899824	Romance	0.482918
Womb	Sci-Fi	Sci-Fi	0.220943	Action	0.178609
2012	Sci-Fi, Disaster	Sci-Fi	0.706881	Disaster	0.581384
Mission Impossible	Action, Spy	Action	0.603631	Martial Arts	0.181121

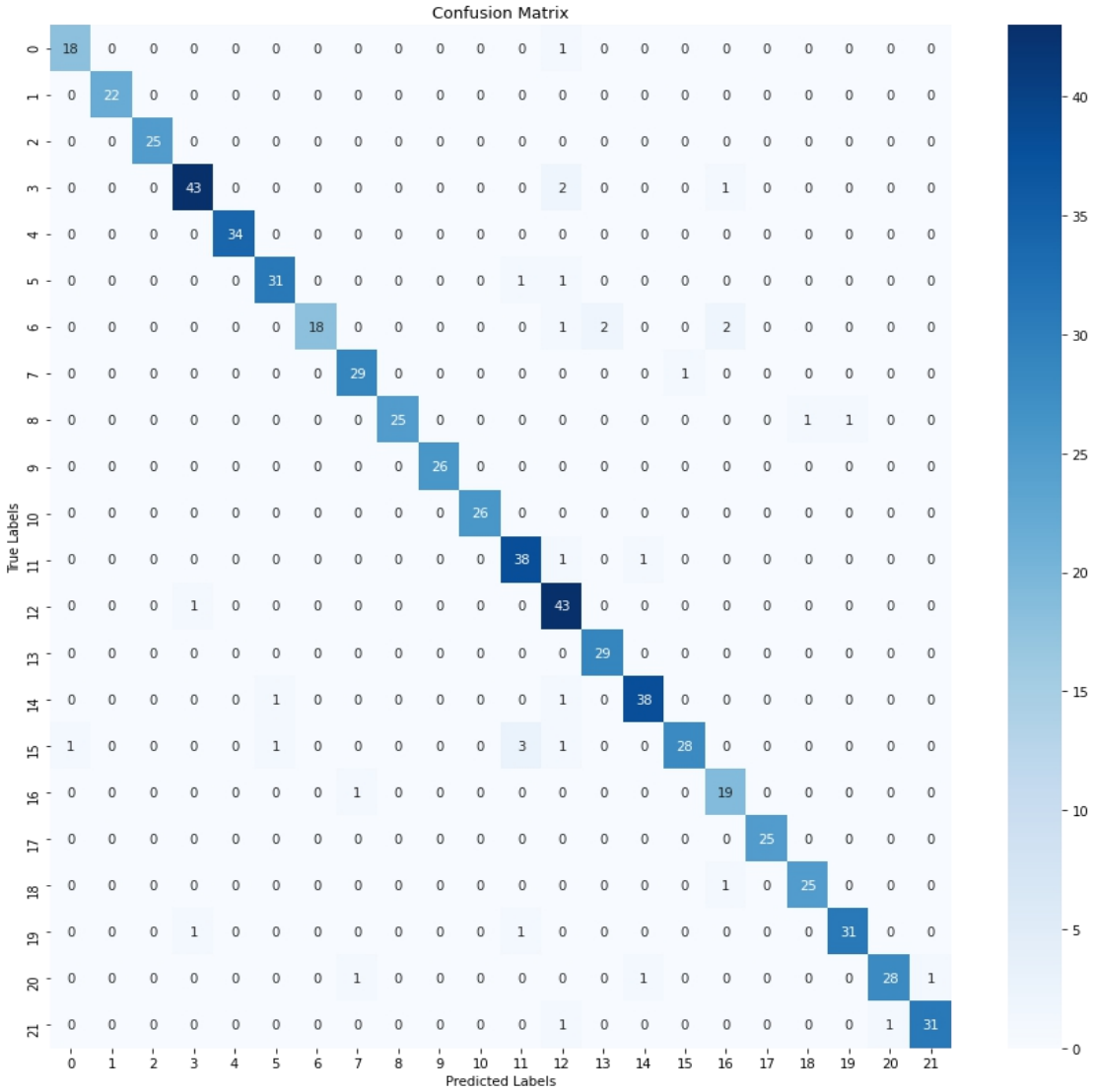




3.2 Movie Subtitles

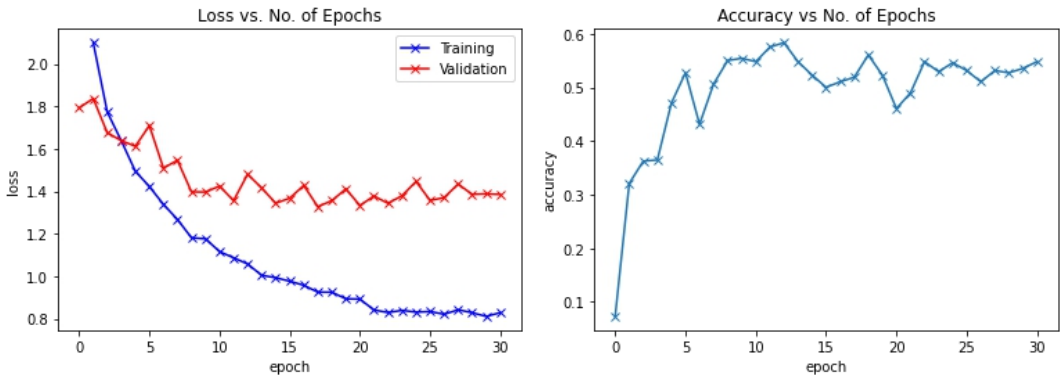
The subtitle classification file achieved an accuracy of 94.89%. Below are and the list of label encodings and the confusion matrix:





3.3 Movie Trailer

The model we trained to classify frame grids extracted from the trailers still has plenty of room for improvement, having an accuracy of about 55%. Below are the results of the training:



3.4 Text Statistics & Facial Expressions

GridSearchCV is used to find the optimal results. We experimented on a number of different models, and based on our experimentations, RandomForestRegressor with hyperparameters shown below yields the best result (72.72% accuracy, able to correctly predict 16 out of 22 test movies).

```
Fitting 3 folds for each of 180 candidates, totalling 540 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:   12.8s
[Parallel(n_jobs=-1)]: Done 158 tasks      | elapsed:   55.6s
[Parallel(n_jobs=-1)]: Done 361 tasks      | elapsed:  2.1min
[Parallel(n_jobs=-1)]: Done 540 out of 540 | elapsed:  3.2min finished
GridSearchCV(cv=3, error_score=nan,
             estimator=RandomForestRegressor(bootstrap=True, ccp_alpha=0.0,
                                             criterion='mse', max_depth=None,
                                             max_features='auto',
                                             max_leaf_nodes=None,
                                             max_samples=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=None,
                                             oob_score=False, random_state=None,
                                             verbose=0, warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'max_depth': [1], 'max_features': [2, 3, 5, 7, 9],
                         'min_samples_leaf': [3, 4, 5],
                         'min_samples_split': [8, 10, 12],
                         'n_estimators': [100, 200, 300, 1000]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=2)
```

3.5 Combining

Below are the combined results from the previous four models. The darker green colour marks the maximum possibility within the row, while the lighter green the second highest possibility. The bold and underlined numbers mark the actual labels for the movie.

NO.	movie	action	adventure	comedy	disaster	horror	martialarts	romance	scifi	spy	ACTUAL_GENRE(S)
0	jumanjiwelcometothejungle	0.095862874	**0.226205623**	**0.122390213**	0.089459386	0.072923997	0.094978076	0.093454111	0.118360526	0.086365195	['adventure', 'comedy']
1	paul	0.09828497	0.095659374	**0.135690251**	0.111211408	0.117888223	0.073291589	0.129217116	**0.128681557**	0.110075513	['comedy', 'scifi']
2	shaolinsoccer	0.104858316	0.086981328	**0.116806017**	0.05125524	0.063234455	**0.357009059**	0.085801135	0.069317887	0.064736563	['comedy', 'martialarts']
3	spectre	**0.130930623**	0.077350895	0.093421217	0.07915614	0.073039191	0.073593893	0.069681839	0.074450265	**0.328375936**	['action', 'spy']
4	sputnik	0.100040047	0.098626727	0.086954673	0.16280216	**0.119952369**	0.082265535	0.077849254	**0.171825651**	0.099683584	['horror', 'scifi']
5	thecore	0.092325488	0.064513513	0.081518785	**0.253474241**	0.120163069	0.063524598	0.081007769	**0.135225458**	0.10824708	['disaster', 'scifi']
6	thediscovery	0.090368508	0.082166394	0.067146528	0.345096361	0.084553028	0.071497071	**0.054268664**	**0.135380139**	0.069523307	['romance', 'scifi']
7	theedgeofseventeen	0.091392015	0.073019633	**0.147530317**	0.080406344	0.128067468	0.082352266	0.217861778	0.093334111	0.086036069	['comedy']
8	thefastandthefurioustokyodrift	**0.120458127**	0.080438964	0.191144837	0.061247654	0.08681984	0.168242558	0.105684124	0.086477963	0.099485934	['action']
9	thefinalmaster	**0.112062177**	0.102762786	0.097214847	0.048513635	0.060400035	**0.35493904**	0.075257776	0.08426221	0.064587494	['action', 'martialarts']
10	thekissingbooth	0.08631245	0.06890932	**0.140630766**	0.065837674	0.101146514	0.133875639	**0.254689627**	0.073484821	0.075113189	['comedy', 'romance']
11	theperfectstorm	0.101013374	0.090414962	0.090865466	**0.184376341**	0.137051312	0.078052396	0.095503217	0.144746878	0.077976054	['disaster']
12	thephotograph	0.100473003	0.068700674	0.106140933	0.11550566	0.154167051	0.078930442	**0.197864544**	0.098822448	0.079395246	['romance']
13	thering	0.086823574	0.085888241	0.069795691	0.264250403	**0.132232728**	0.074546816	0.084768665	0.136963089	0.064730795	['horror']
14	theseventhadventuresofsinbad	0.088082379	**0.156975482**	0.079022151	0.217547124	0.070597599	0.075957019	0.066169912	0.175121649	0.070526684	['adventure']
15	thismeanswar	0.109314758	0.066420036	**0.177217147**	0.076511739	0.130841919	0.09131296	0.174837339	0.07908694	**0.094457162**	['comedy', 'spy']
16	triplefrontier	**0.172907202**	**0.072748479**	0.129087941	0.09926126	0.145670763	0.096944296	0.07273041	0.097042414	0.113607235	['action', 'adventure']
17	tronlegacy	**0.117640849**	0.087513044	0.126590966	0.173560006	0.11333728	0.117671013	0.066237077	**0.101546191**	0.095903574	['action', 'scifi']
18	vampiresvsthebronx	0.125437612	0.065708692	**0.189924675**	0.063582589	**0.179556921**	0.109869419	0.098724803	0.073802	0.093393289	['comedy', 'horror']
19	womb	0.090481796	0.086517375	0.095130768	0.106992702	0.161433614	0.082489756	0.125501524	**0.180473366**	0.0709791	['scifi']
20	worldwarz	**0.121985782**	0.079231465	0.081198752	0.181231758	**0.171154841**	0.068021563	0.072223875	0.114236637	0.110715328	['action', 'horror']
21	zoolander2	**0.106041873**	0.110309672	**0.184007618**	0.077324114	0.103835655	0.110196622	0.137488907	0.074320358	0.09647518	['action', 'comedy']

Out of 22 test movies, 10 are 100% correctly predicted (correctly predicted 2 out of 2 genres, or correctly predicted 1 out of 1 genre), 8 are 50% correctly predicted (correctly predicted 1 out of 2 genres), and 4 are not predicted correctly (correctly predicted 0 out of 2 genres or 0 out of 1 genre).

4 CONCLUSION

Although vastly imperfect, our methods have shown to be able to correctly predict the genres of a number of movies, of a number of different genres, with an accuracy of around 82%. In the future, we hope to be able to improve our existing models, and/or continue to make a model that could further summarize movies into shorter, textual summaries via abstractive summarization.

5 REFERENCES

1. D. Jacob, C. Ming-Wei, L. Kenton, and T. Kristina. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Computation and Language NAACL-HLT*.
2. S. Muhammad, U. Amin, A. Jamil, A. Naveed, R. Seungmin, and B. Sung Wook. (2018). Integrating Salient Colors with Rotational Invariant Texture Features for Image Representation in Retrieval Systems. *Multimedia Tools and Applications*, pp.77.
3. C. Shih-Fu, and S. Hari. (2000). Structural and Semantic Analysis of Video. *Multimedia and*

Expo ICME, (2).

4. M. Karin, G. Nagia, and I. Mohamed. (2013). Unsupervised Video Summarization via Dynamic Modeling-based Hierarchical Clustering. *12th International Conference on Machine Learning and Applications*.
5. R. Anyi, X. Linning, X. Yu, X. Guodong, H. Qingqiu, Z. Bolei, and L. Dahua. (2020). A local-to-Global Approach to Multi-modal Movie Scene Segmentation. *Computer Vision and Pattern Recognition*.
6. Haq, U. I., Muhammad, K., Hussain, T., Kwon, S., Sodanil, M., Baik, S. W., Lee, M. Y. (2019). Movie Scene Segmentation using Object Detection and Set Theory. *International Journal of Distributed Sensor Networks*, (15).
7. Lee, S., Kim, I. (2018). Multimodal Feature Learning for Video Captioning. *Mathematical Problems in Engineering*.