

NEURAL NETFLIX

CS109b

Leonard Loo

Kimia Mavon

Nikhila Ravi

Yihang Yan

Background

In this paper, we explore a variety of data analyses by utilizing visualizations, categorical, and continuous data to identify features that predict genres. Data was gathered from two movie databases, The Internet Movie Database (IMDb) and The Movie Database (TMDb). Each movie included in both IMDb and TMDb contain the same external identification number, allowing the two databases to be matched.

The outcome variable to predict is movie genre. The predictor variables includes a variety of fields described below, including categorical and continuous data, and image data. This paper discusses the analysis conducted, including a multi-label prediction on each movie using both traditional machine learning techniques and deep learning.

Data

Data crawling: To use the API, we made a request to the TMDb and IMDb remote web servers and retrieved the data we needed using the Python requests. From TMDb, we called the API for 'id', 'genre_ids', 'poster_path', 'title', 'overview', 'release_date', 'popularity', 'original_title', 'backdrop_path', 'keywords', 'vote_count', 'video', 'adult', 'vote_average', 'original_language' and movie posters using the poster url. Because TMDb limits 4 requests per second, a timer was implemented. After creating a .CSV of this data, we created a binary indicator for TMDb's nineteen genre IDs across columns. Next, we matched the IMDb IDs using the TMDb IDs provided in the .csv file generated. We then called IMDb's available features including 'imdb_id', 'director', 'imdb_votes', 'certificate', 'num_stunts', and 'num_fx'. We then merged these CSV files and output a compiled file of about 7,000 movies.

Feature engineering: Using data crawled from the API, we focused on the following:

- Float values (popularity, vote_count, vote_average, imdb_votes, num_stunts, num_fx)
- release_date: this was parsed to years and months, and again parsed into decades. Movies made in a similar period are likely correlated, and differ more from decade to decade. This methodology is practiced in song classification techniques.
- certificate: This categorical variable was manually reduced to those used most commonly: Unrated, U, PG, 12, 15, 18 and R.
- director, keywords: As discussed later in our random forest analysis, this data proved less fruitful for analysis.

We combined different genres to create 7 different groups based on the correlation between each genre pair to avoid the collinearity. A correlation matrix of these genres (Figure 1) is located in the appendix .

Group 1 - War, History

Group 3 - Fantasy

Group 6 - Science Fiction,

Group 2 - Crime, Mystery,

Group 4 - Family, Animation

Action, Adventure

Thriller, Drama, Horror

Group 5 - Romance, Music

Group 7 - Comedy

Sampling: The genre groupings were heavily unbalanced. Each of the 7 genre groups labels contained the following number of movies: 620, 5506, 982, 876, 1762, 3502, and 2969. Therefore, we developed our own multi-label over/under-sampling algorithm. We over-sampled the minority until each minority label aggregated to the mean (2,316 movies). We then under-sampled the majority to half of the mean (1,158 movies). Because movies contain more than one label, over-sampling the minority will retrieve more counts of the majority label as well, so the majority will need to be compensated by heavily under-sampling. After our sampling algorithm, genres across the seven labels included more balanced counts: 2739, 8482, 4048, 3861, 4615, 6596, 6173.

Evaluation Metrics: We used three multi-label evaluation metrics to evaluate the performance of our multi-label classification, as a film may be categorized as more than one genre.

- The Hamming loss estimates the fraction of the wrong labels to the total number of labels. It is used in order to control the forgiveness of errors that are made in predicting labels as it can, in principle, be

minimized without taking label dependence into account. It is important to note that a lower hamming loss demonstrates a better performance.

- Percent exact estimates proportion of movies for which at least one genre was predicted correctly
- Percent at least one estimates the percentage of movies that have all their genres correctly predicted

Exploratory Data Analysis

Selection of Keywords by Random Forest: We used One vs Rest Classification to predict all labels for a new sample for which the respective classifiers predict a positive result. This was originally proposed as a potential dimensionality reduction technique, by selecting the top keywords from the feature importances for each genre and add them to the movie metadata. We first generated binary indicator matrix for the unique ~12,000 keywords and use a separate random forest classifier for each of the seven genres to determine which keywords are most relevant in classifying each genre. Top keywords for each genre were computed, as shown in table X, in the appendix. This analysis produced poor results. The plots of feature importance have very high variance/error bars. Additionally, the random forest on keywords computed a high hamming loss (.19) indicating higher fraction of mislabelling and also has low percent exact match (27.5%). In conclusion, random forest is not a good fit for the analysis.

Metadata models

SVM One-v-rest & RF one v rest: We used One vs Rest Classification to independently train one binary classifier for each label using support vector machines (SVM) and random forest (RF) models. We predict all labels for a new sample for which the respective classifiers predict a positive result. The untuned SVM yielded a hamming loss, percent exact, and percent at least one of 0.085, 62.2%, and 89.16%, respectively. The untuned RF yielded a hamming loss, percent exact, and percent at least one of 0.09, 60.6%, and 94.7%. Then, the parameters of SVM and RF were tuned using cross validation. The performance of tuned models is shown in Table 1. Comparing the untuned radial SVM to the tuned, the hamming loss went down slightly and the percent exact went up to 63%. However, it seems the percent of at least one genre matching dipped slightly. After tuning, all error measures improved.

ANN (Sigmoid, >0.5 classifies as 1): We also explored deep learning (ANN) to reduce the error rate for multi-label classification. We scaled all movie metadata features and hand tuned the parameters for the model. From a hand-tuned ANN, we can see a sharp decline in performance based on Hamming loss or percent exact. This decline is likely due to the standardization of features before we run ANN. When we standardized before the other models, we also degraded in performance. It is possible that the difference between features is actually a good predictor. Fortunately with ANN, we see better performance for at least one percent.

Model comparison: Comparing the tuned radial SVM, the tuned random forest, and the ANN we found that the tuned Radial SVM has the lowest hamming loss and highest percentage of exact matches, indicating a low error rate. The tuned RF also has similar performance to the radial SVM with higher percent of at least one label matching without much decrease in the percent exact match. The random forest on the keywords has a higher hamming loss indicating higher fraction of mislabelling and also has low percent exact match. Interestingly, SVM better predicted exact matches, while RF is superior in predicting percent at least one. However, the untuned ANN on the movie metadata has high percent of at least one label matching though it has very low percent exact match and the highest hamming loss out of all the models due to the standardization.

Model	hamming loss	percent exact match	percent at least one match
Tuned Radial SVM on movie metadata	0.082	63.05%	88.8%
Tuned RF on movie metadata	0.088	61.5%	95.3%
ANN on movie metadata	0.248	13.1%	96.8%
RF on keywords	0.194	27.5%	75.2%

Table 1. Performance Matrix of all metadata models

Image data models

PCA on posters with SVM: Our baseline image data model used the movie posters for multilabel classification by applying dimensionality reduction to the image pixel vectors. After transforming the images to grayscale, PCA was applied. The top PCs that contribute to 90% of the data is 166. The training and testing data were projected onto the retained principal components and the reduced feature space was used as the input into a OneVsRestClassifier using SVM/RF.

CNN from scratch: We fitted our CNN on 7,814 movie posters with balanced genre labels. We then tested our CNN on 3,700 movie posters. To reduce computation time, we diminished our posters to 148x100 pixels to maintain the aspect ratio of 741:500.

Our initial CNN includes:

- A convolutional layer with 32 filters, (3,3) kernel size, ReLU activation, Max Pool layer with (3,3) pool size
- A convolutional layer with 32 filters, (3,3) kernel size, ReLU activation, Max Pool layer with (2,2) pool size
- A convolutional layer with 64 filters, (3,3) kernel size, ReLU activation, Max Pool layer with (2,2) pool size
- A fully connected network with 1 hidden layer with 64 neurons and ReLU activation
- A sigmoid output, categorical entropy loss function, adam optimizer, SGD batch size of 16, and 20 epochs

Though 20 epochs appears to be too few, further computation demonstrated that 20 epochs is sufficient because our loss descended the most within the first 10 epochs. Our performance evaluation yielded a hamming loss of 0.153, percent exact of 43.9% and percent at least one of 94.8%. The reported hamming loss and percent exact were worse than using metadata to classify (0.094 and 60.6% respectively), but percent at least one maintained a similar performance.

Tuned CNN from scratch: Next, we used grid search, a model hyperparameter optimization technique to construct and evaluate one model for each combination of parameters. We experimented with the following different configurations:

- SGD Batch size: 16 or 32
- Epochs: 20 or 50
- Optimizer: Adam or rmsprop
- Loss function: Binary or categorical cross entropy
- Number of filters in each Conv layer – 68 or 128 (with the last convolutional layer's filters doubled). We also added a dropout of 0.2 to each layer.

Though the Grid Search was computationally intensive, we learned: 1) Additional hidden neurons/filters with dropout is more effective than fewer neurons/filters, 2) Adam and rmsprop optimizers yield similar performance, 3) Categorical cross entropy slightly outperform binary cross entropy in multi-label prediction, 4) The number of epochs past 20 does little to improve performance. Approximately 20 epochs is sufficient given our small dataset.

Using this, we redesigned our CNN with the following characteristics:

- A convolutional layer with 64 filters, (3,3) kernel size, ReLU activation, Max Pool layer with (3,3) pool size, with 0.2 dropout
- A convolutional layer with 64 filters, (3,3) kernel size, ReLU activation, Max Pool layer with (2,2) pool size, with 0.2 dropout
- A convolutional layer with 128 filters, (3,3) kernel size, ReLU activation, Max Pool layer with (2,2) pool size, with 0.2 dropout
- Fully connected network with 1 hidden layer with 128 neurons and ReLU activation, with 0.2 dropout
- Sigmoid output, categorical entropy loss function, adam optimizer, SGD batch size of 16, and 20 epochs

Our performance then increased to a hamming loss of 0.131, percent exact of 49.8% and percent at least one of 95.7%. This is an improvement in all performance metrics compared to the untuned CNN from scratch.

CNN with data augmentation: Data augmentation is often used to prevent over fitting in cases of data scarcity. Data augmentation was performed on 500 images, to improve prediction results on a small sample. Compared with the basic CNN model from scratch without augmentation, data augmentation is able to reduce hamming loss from approximately 0.15 to 0.11, increase percent exact from 0.44 to 0.47, and increase percent at least one to 100%.

VGG16 features + ANN: To leverage results and computational power from prior work, a VVGnet model (Simonyan & Zisserman, 2015) with pre-trained weights from Imagenet was used as a feature extractor. This model is available through the keras api. The VVG16 network was truncated before the fully connected output layer, preserving only the convolution/max pooling layers.

First, the testing and training data was passed through the network to extract the output feature vector for each image. The feature vectors were fed as the input into an MLP consisting of two fully connected layers with ReLU activation each followed by 50% dropout and a final layer with 7 units producing a 7 dimensional binary vector to predict genre labels.

VGG16 retrain last layer: In this approach, the VVGnet model was frozen up until the the last convolutional block which was kept trainable. A custom fully connected output block was attached at the end, consisting of two fully connected layers with ReLU activation each followed by 50% dropout and a final layer with 7 units producing a 7 dimensional binary vector.

In this model, the weights of the last convolution block of VVGnet are also being updated by backpropagation as opposed to the first model in which VVGnet was simply used for feature extraction. The fine tuned model was trained for 50 epochs with 'adam' as the optimizer and a batch size of 50 and evaluated using the 'binary cross entropy' loss.

Analysis and Challenges: A comparison of the loss measures for each model is shown in Table 2.

model	hamming loss	percent exact match	percent at least one match
Deep network from scratch	0.154	43.8%	94.8%
Deep network from scratch + parameter tuning	0.13	49.7%	95.7%
Pretrained model + MLP	0.099	56.08%	95.9%
Pretrained model + fine tuning of last convolution block + MLP	0.094	57.3%	96.8%

Table 2. Comparison of loss measures for CNN models trained on image posters

Each of the models was trained on the entire training set and used to make predictions on the testing set. Parameter tuning of the basic CNN model resulted in the hamming loss decreasing from 0.15 to 0.13, percent exact increasing from 0.44 to 0.50, and percent at least one increasing slightly. We did not expect to achieve significantly greater improvements through further fine tuning of parameters so did not pursue this model further.

Extracting features from the pretrained VGGnet model and feeding into a custom MLP, resulted in a 0.06 decrease in the hamming loss and 10.3% increase in the percentage exact match. This is marked improvement in performance. Further fine tuning of the last convolution block only resulted in a 1% increase in the percentage exact match and 0.005 drop in the hamming loss. We speculate that further improvements in performance could have been observed if we had carried out the training for larger number of epochs and modified the parameters of the fully connected output layer. In particular we could have trained the fully connected layer first using feature vectors as inputs (model 3 in Table 2)), saved the weights of the MLP and for model 4, initialised the network with the pre-trained weights. With this approach, in model 4, the output layer weights would not need to be initialised randomly and gradient updates would be less extreme, leading to finer tuning of the weights of the last convolutional layer of VGGnet.

Some challenges in the use of deep models included:

- Large image sizes causing memory issues on AWS (the resolution had to be decreased to enable the models to be trained).
- Slow loading of images from external storage (we chose to use github), forcing us to limit the size of our training data to avoid having to wait long periods to load data before training, and enable us to iterate faster with our models.

Exploratory Idea

In our additional exploration, we read the popular paper 'A Neural Algorithm of Artistic Style' (Gatys et al., 2015 - <https://arxiv.org/abs/1508.06576>) that demonstrates how to use neural networks to transfer artistic style from one image onto another. This is the underlying technology behind the popular Prisma app. We adapted our code to transfer styles between different genres of posters. For example, here we made Finding Dory, a child-preferred genre into a horror movie by using a bloody style image:



We do this in a series of steps:

- 1) Preprocess images:
 - a. Resize content and style images to the same size.
 - b. Add an extra dimension for processing later.
 - c. Standardize pixel values with respect to VGG16 mean values
 - d. Define keras variables and initialize combination image (a third image which is the final output image after mixing content and style images).

2) Load VGG16: The core idea introduced by Gatys et al. (2015) is that CNNs pre-trained for image classification already know how to encode perceptual and semantic information about images. We use this method with the feature spaces provided by VGG16 to independently work with content and style of images. Because we're not interested in the classification problem, we don't need the fully connected layers or the final softmax classifier. We only need the 16 layers in the 'D' column of VGG16. The crux of the paper we reproduce is that the style transfer problem can be posed as an optimization problem, where the loss function, if optimally minimised, can be decomposed into three distinct parts: the content loss, the style loss, and the total variation loss.

3) Define loss functions:

- a. Content loss: Content loss is the (scaled, squared) Euclidean distance between feature representations of the content and combination images. We draw the content feature from: block2_conv2 (as stated in the paper).
- b. Style loss: To define style loss we first define the gram matrix, which is proportional to the covariances of corresponding sets of features, and thus captures information about which features tend to activate together. We then compute the style loss, which is the (scaled, squared) Frobenius norm of the difference between the Gram matrices of the style and combination images. Style loss is a bit more complicated, and we draw the features from multiple feature layers as mentioned in the paper (block1_conv2, block2_conv2, block3_conv3, block4_conv3, block5_conv3).
- c. Total variation loss: Instead of just using style and content loss, we add a total variation loss to regularize content and style loss to encourage spatial smoothness. We then combine the style and content loss, with this variation loss in a weighted fashion. The weights of each loss define the trade off between preserving the objects in the image and transferring the style.

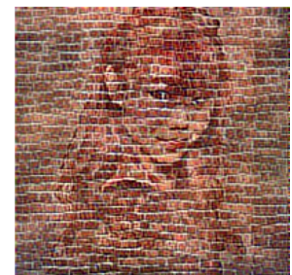
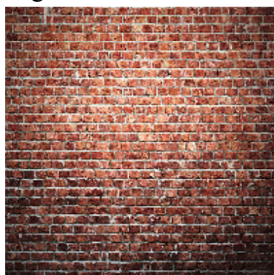
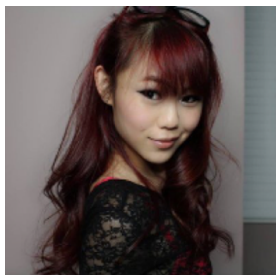
4) Optimization:

- a. The combination image is initialized as a random collection of pixels, and we use the L-BFGS algorithm (a quasi-Newton algorithm that's significantly quicker to converge than standard gradient descent) to iteratively improve upon it, while transferring both the content image and style image to it based on the loss functions.
- b. We run the above gradient descent algorithm for 10 iterations and got the above results. Each iteration takes about 28 seconds on a GPU, and could be sped up to 2 seconds if we resize the images to 200x200 instead of using the original 500x741.

There are several parameters which we hand-tuned to arrive at a satisfactory combination movie poster showcased above including:

- Ratio of content and style loss: the papers suggested ratios, so we played around with these.
- Which layers of VGG16 to use for the content and style loss functions: different papers recommended different layers, so we tested out different layers.

We also learned that the utmost importance to achieve a beautiful combination image much like Prisma's is to use style images that have one significant feature, such as a brick wall, or sea waves. Here's what happens when we choose a brick wall as a style image:



Conclusion

This paper summarizes methods used to build an effective image classifier using very few training examples reduced dimension images. Further data analyses would include training for larger number of epochs, modifying the parameters of the fully connected output layer, and adding a more comprehensive training dataset. We believe including these edits to future analysis will strengthen genre prediction performance metrics.

References

1. Very Deep Convolutional Networks for Large-Scale Image Recognition, Karen Simonyan, Andrew Zisserman, ICLR 2015

Databases

1. The Internet Movie Database (<http://www.imdb.com>). Used with permission.
2. The Movie Database (<https://www.themoviedb.org>). Used with permission.

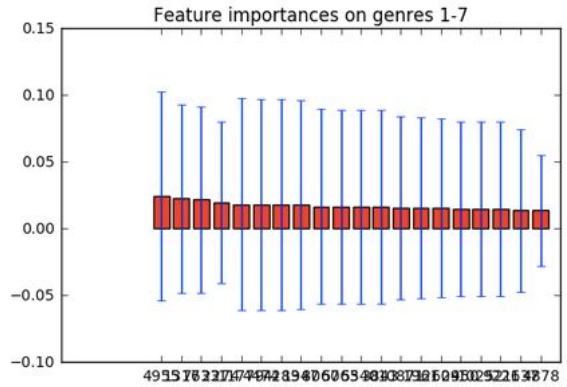
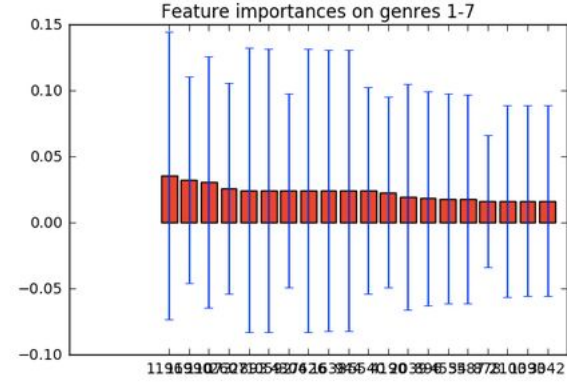
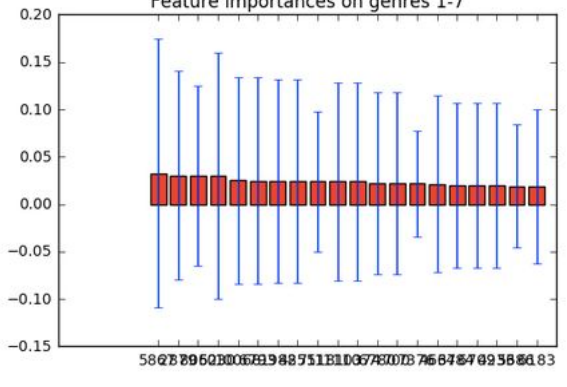
Appendix

Figure 1: The correlation matrix of movie genre pairs



Table 1: Random forest analysis on keywords predicting genre.

Genre ID	Genre	Keywords	Feature Importances
1	"War" "History"	camouflage.uniform exotic.dancer beauty.queen.contest deputy construction.worker	

2	"Crime" "Mystery", " Thriller" "Drama" "Horror"	graduate.school booze.cruise mud.wrestling cockney.accent time.warp	 <p>Feature importances on genres 1-7</p>
3	"Fantasy"	travel.writer treehouse stephen.king covert.operation dead.body	 <p>Feature importances on genres 1-7</p>
4	"Family" "Animation"	inspired.by.real.stories dating.woes oberflächlichkeit groundhog.day schoolboy	 <p>Feature importances on genres 1-7</p>

5	"Romance" "Music"	simulation hospital.room solar.system living.alone electronic.voice.phenomena	
6	"Science Fiction" "Action" "Adventure"	contract.killer gay.vampire post.war man.woman.relation psychoanalysis	
7	"Comedy"	buried.treasure detour home.movie hostage.killed montreal	