Michael Siebel
Final Project: Group 5
DATS 6203
Machine Learning II
Section 10

Game of Classification
Deep Learning on Game of Thrones Screenshots

Group Report

Introduction

Game of Thrones was the television network HBO's highest rated and most profitable show. Airing from 2011-2019, it recently had its series finale, but remains an important show on HBO's online streaming platform HBO Go. Besides being highly viewed, the show was at one time the most pirated television show in history.

A common issue television networks face is improper distribution of their intellectual property. This project seeks to create a copyright detection algorithm that can identify whether an image belongs to Game of Thrones or a television show from a different genre. As such, the algorithm will have a binary classification outcome.

Objective

Game of Thrones is known for having numerous of characters and locations. As a result, the goal of the algorithm cannot be to identify character faces but rather other attributes. The show is fantasy-based, existing in a fictional, medieval inspired land. It is a costume-drama largely set outdoors or in rustic castles and ancient huts. This aesthetic would be easy for a person, even one without knowledge of the characters and plot, to identify. Because it does not contain many consistent objects, though, it will likely be difficult for a machine.

The attributes these photos have include high production values, color-treatment, and an ancient setting. The high production values are realized in the details of the photos. Complex costumes, expansive sets, and large computer-generated creatures set Game of Thrones apart from other shows. Its color treatment means that colors tend to be consistent within scenes. For example, cold locations have blue colors emphasized and warm locations have orange colors emphasized via post-production. Therefore, red-blue-green (RGB) values capture detail, but not necessarily wide-ranging colors. Finally, the ancient setting is both mystical and serious in tone—something easy for a human to recognize but with little explicit attributes for a machine to identify.

As Game of Thrones has complex images, the rest of the data will be comprised of shows from a very different genre—13 comedic sitcoms spanning from 1989-Present. The shows included Arrested Development, The Big Bang Theory, Boy Meets World, Fresh Off the Boat, Fresh Prince of Bel-Aire, Friends, How I Met Your Mother, Kim's Convenience, It's Always Sunny in Philadelphia, Saved by the Bell, Seinfeld, and Spin City. Sitcoms have closer to real world aesthetics, with characters wearing modern clothing and being in modern, often indoor settings. They also have lower production values, meaning there is likely less detail in the images. They are rarely color treated and tend to be colorful, meaning the RGB values will vary more in any given cluster of pixels.

Data

All images were web scrapped from the Internet Movie Database (IMDB.com) using the statistical software R. All images are screenshots and therefore do not include behind the scenes or promotional photos. Unfortunately, I found it difficult to web scrap the original-sized images and instead was only able to collect the thumbnail images. Therefore, all screenshots are pre-cropped to 100x100 pixels, causing the images to be of low quality. This created an additional challenge for classification, although successful classification of a low quality image likely translates better to classification for high quality counterparts possibly increasing the robustness and usefulness of this copyright detection algorithm.

In total, there are 3,883 Game of Thrones screenshots and 9,653 sitcom screenshots for a total of 13,535 images. The data is somewhat imbalanced with Game of Thrones comprising only around 30% of the total data. The dataset is large enough to necessitate a deep network.

Modeling

The primary modeling was done with a convolutional neural network (CNN) as it accepts matrix input without flattening, retaining the images' RGB values. This model was chosen as the color scheme was anticipated as important to identify key Game of Thrones attributes. Overall, image filters were expected to capture an image's general aesthetic better than the neurons of a Multilayer Perceptron (MLP).

The data was first split into training (70%), testing (15%), and validation (15%). The models used a binary cross entropy loss (BCELoss) function for training as it is designed for binary outcomes. All tuning of parameters were evaluated on the testing data using F1-scores. Once a high performing network architecture was found, a final model was evaluated on the validation data, again, using F1-scores.

F1-scores provide a harmonic average between precision and recall. Precision is often a more valuable metric for judging the performance of binary data in which the predicted class is under sampled. However, precision does not account for false negatives, which can still be an issue as my data is only a little imbalanced. Therefore, F1-scores provide balance in evaluation. When models contained similar F1-scores, the model with the higher number of true positives was chosen as the better. All coding was done using Pytorch.

Data Augmentation

Given the imbalances of the dataset, all Game of Thrones images were oversampled by selecting them twice within the training set. Upon selecting each image twice, the images and corresponding labels were shuffled (see Figure 1 for sample syntax).

Figure 1. Syntax for Oversampling Game of Thrones Images

```
# Oversample GoT
x_ovsp = []
y_ovsp = []
for im, lab in zip(x_train, y_train):
    if lab == 1:
        x_ovsp.append(im)
        y_ovsp.append(lab)

y_ovsp = np.array(y_ovsp)

def shuffle_train(a, b):
    assert len(a) == len(b)
    shuffled_a = np.empty(a.shape, dtype=a.dtype)
    shuffled_b = np.empty(b.shape, dtype=b.dtype)
    permutation = np.random.permutation(len(a))
    for old_index, new_index in enumerate(permutation):
        shuffled_a[new_index] = a[old_index]
        shuffled_b[new_index] = b[old_index]
    return shuffled_a, shuffled_b

x_train = np.concatenate((x_train, x_ovsp), axis=0)
y_train = np.concatenate((y_train, y_ovsp), axis=0)
x_train, y_train = shuffle_train(x_train, y_train)
```

Table 1 shows that this process increased the representation of Game of Thrones images from 29% to 44% of all images in the training set.

Table 1. Oversampling Game of Thrones Training Images

Labels	0 Sitcom	1 Game of Thrones
Pre-Oversampling	6763	2711
Post-Oversampling	6763	5422

All images were then resized to 50x50 pixels—half their original size. Reducing image size to such a small pixel density decreasing the amount of pooling that can occur. However, the smaller the resize, the greater the data augmentation effects as its image territory became more distinct from a later batch. The data augmentation randomly effected each photo via cropping, color jitter, flipping, rotating, and normalization. Given that Game of Thrones images were present twice per training epoch, the data augmentation effects need to be impactful in order for the oversampled image to be treated as distinct from its original counterpart. In other words, oversampling does not mean training on the same photos in the same manner during each training epoch. On the test and validation datasets, images were only resized, center cropped, and normalized—avoiding randomized transformations or oversampling in order to maintain the original data structure.

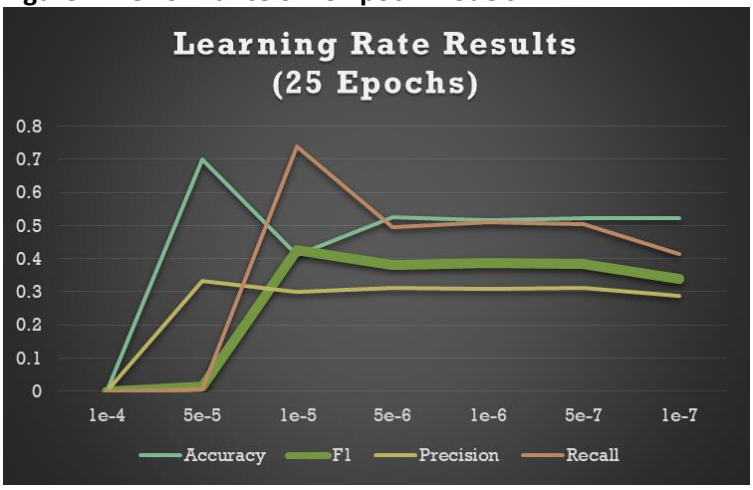
Training

The training process began with a five convolutional layer network with one fully connected layer followed by the output fully connected layer. The images could only be down-sampled, during the pooling process, five times due to their small size. Each layer contained batch normalization; there were three 20% dropouts (on convolutional layers 3 and 5 and the fully connected layer) in order to reduce the likelihood of overfitting. Finally, each model contained oversampling.

The training process primarily focused on the learning rate. Previous experience on Exam 1 and Exam 2 indicated that lower learning rates translated the performance on the training data much better to testing data.

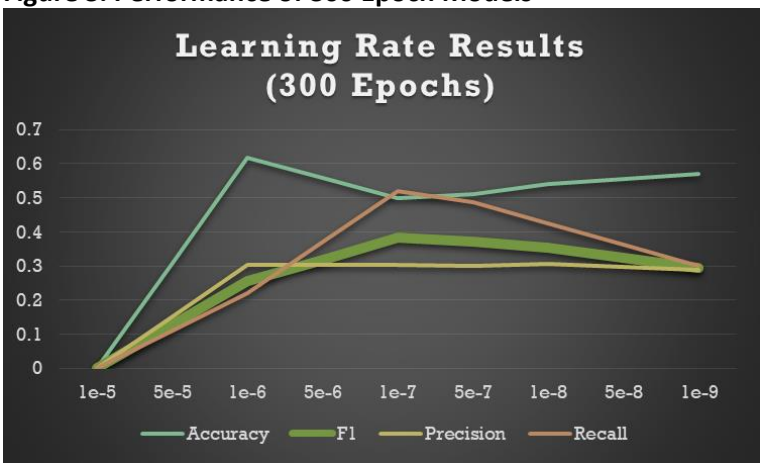
Eight models were initially run with different learning rates ranging from 1e-2 to 1e-7. Only after the learning rate decreased to 5e-5 did the model begin predicting any Game of Thrones images. From a learning rate of 1e-5 to 1e-7, the F1-score did not change greatly. The highest F1-score was with a learning rate of 1e-5, although this may be due to the random shuffling of each batch producing more favorable results by chance. In other words, a repeated run with this learning rate (with a different sequence of images per batch) may produce a lower score. Figure 2 shows the results of these models.

Figure 2. Performance of 25 Epoch Models



Next, the models were updated to run for 300 epochs. The best learning rates were lower than when the models ran with 25 epochs. A learning rate of 1e-7 achieved the highest F1-score. Notably, learning rates from 1e-6 through 1e-9 contained the same precision scores. The recall rate increased as the learning rate lowered to 1e-7 and decreased as the learning rates lowered past 1e-7. Overall, the best F1-score possessed 314 true positives out of a test dataset of 2,030, while the false positives were 725. This indicates that oversampling has allowed the model to over predict Game of Thrones images. Finally, it's important to note that the F1-scores were not higher than the 25 epoch model runs. Figure 3 shows the results of these models.

Figure 3. Performance of 300 Epoch Models



After a final process of tinkering, the final model architecture was slightly tweaked as it updated to using 500 epochs. The decision to increase epochs to 500 was made because, although this did not seem to

improve the F1-scores, increasing epochs did result in higher numbers of true positives. The new model contained more dropouts in hopes that this would translate the training scores better to testing and validation datasets. This involved adding two more 20% dropouts (to convolutional layers 2 and 4) for a total of five 20% dropouts. Finally, a 1e-8 learning rate was selected. It received a similar F1-score but contained a few more true positives. Table 2 displays the main model's network architecture.

Table 2. Main Network Architecture

	Input	Output	Activation	Batch Normalization	Pooling	Dropout
Convolutional Layer 1	3	16	ReLU	2D	Max Pooling	
Convolutional Layer 2	16	32	ReLU	2D	Max Pooling	0.2
Convolutional Layer 3	32	64	ReLU	2D	Max Pooling	0.2
Convolutional Layer 4	64	128	ReLU	2D	Max Pooling	0.2
Convolutional Layer5	128	256	ReLU	2D	Max Pooling	0.2
Fully Connected Layer	256	128	ReLU	1D		0.2
Output layer	128	1	Sigmoid			

The model was run on the validation dataset, achieving an F1-score of 0.39 and an accuracy of 0.44. It contained the highest amount of true positives (compared to prior models) of 366. However, to achieve this, the final distribution greatly over predicted the main class, predicting 1,296 Game of Thrones images when there was 568. Table 3 displays the confusion matrix and final distributions.

Table 3. Confusion Matrix and Final Distributions

Confusion Matrix	Predicted Sitcom	Predicted Game of Thrones
Actual Sitcom	533	930
Actual Game of Thrones	202	366

Alternative Models

Given that the main model was generally a poor classifier, I attempted to experiment with three alternative models. One removed oversampling, another reduced the network architecture to three convolutional layers, and the last converted the main model to an MLP version. All alternative models contained a learning rate of 1e-8, were run for 500 epochs, and were evaluated against the validation dataset.

A) The non-oversampling model

This model was otherwise the same as the final model, containing five convolutional layers, five 20% dropouts, etc. With the exception of oversampled images, it contained the same data augmentation procedures of the main model. Its final results reduced the F1-score by 0.1 points to 0.29. However, while the F1-score dramatically declined, so did the number of false positives, leading to a higher accuracy of 0.60.

B) The three convolutional layer model
This model contained three 20% dropouts and maintained the fully connected layer with batch normalization before the output layer. The model contained the same data augmentation procedure of the main model. It achieved similar results to the main model with a slightly lower F1-score (0.35) and a slightly higher accuracy (0.53).

C) The MLP model
This model contained six linear layers with five 20% dropouts. All layers contained 2,500 neurons. The images were converted to grayscale and flattened. Other than featuring a color jitter, the same data augmentation procedure of the main model was used before flattening. Surprisingly, the MLP version had similar classification scores as the main model. It achieved the same F1-score (0.39) and a slightly higher accuracy (0.47). However, it did contain slightly less true positives (342).

Conclusion
I am disappointed by my copy right detection algorithm. The process I took did not seem to optimize it to the degree I was expecting. Overall, I learned four main points during this process.

1) Tuning the learning rate did not affect the results as much as I expected. This was the most tuned parameter I focused on during my project. It is often considered the most important hyper-parameter in modeling (a). I believed that extending it would solve the majority of my complex data concerns. However, in my final model, my loss function did not decline in any meaningful manner across epochs, despite reducing the learning rate. This presumably was due to a low learning rate "getting stuck" in local minima.

2) Extending the number of epochs did little. As the epochs increased, I decreased my learning rate. This did not improve F1-scores but did increase the quantity of true positives as the model began overpredicting Game of Thrones images.

3) The MLP model performed similar to the CNN. Much of my initial theory assumed that the RGB values would be key. However, an MLP architecture using a grayscale presumably focused more on the complexity and details of the photos, such as clothing texture/wrinkles, shadows, and density of set objects. Perhaps my model, being limited by its simplicity, was only evaluating these details and not color scales. CNN's are more efficient with large images. Perhaps because the image size was small, its was manageable by MLP architecture, where the full sized versions would not be.

4) Finally, higher number of layers did appear to improve the F1-score. Using three convolutional layers weakened the F1-score compared to the five convolutional layer comparison. Given the size of images, I could only have a maximum of five pooling layers before I could no longer down-sample. Where convolutional layers are limited, adding additional fully connected layers may increase the complexity where convolutional layers are not possible. Of course, using ensemble modeling could further increase the complexity.

Overall, the model is not good enough to provide use in copyright detection. Oversampling and low learning rates were necessary, but more needs to be done to increase its F1-score.