Convolutional Neural Networks for Movie Poster Image Recognition

Royal Wang
University of Virginia
School of Data Science
rjw8ng@virginia.edu

Brennan Danek
University of Virginia
School of Data Science
bd4bk@virginia.edu

Theodore Thornmann
University of Virginia
School of Data Science
nbx5kp@virginia.edu

***Abstract*—Convolutional Neural Networks (CNNs) are widely used in image recognition tasks, including movie poster recognition. Movie posters are an important visual element in movie promotion, and contain many different features useful for classifying the genre of a movie.**

An approach to using CNNs for movie poster recognition involves using a pre-trained CNN model such as ResNet or DenseNet to extract features from the images. These models have been trained on large image data sets and can recognize basic features such as lines, curves, and colors. The extracted features are then fed into a classification model, such as a fully connected neural network, to predict the genre or box office success of the movie based on the poster image.

To improve the accuracy of the model, data augmentation techniques can be used to artificially increase the size of the training data set. Other techniques such as transfer learning, fine-tuning, and ensemble can also be used to improve the performance of the model. Transfer learning involves using a pre-trained CNN model as the starting point for training a new model on a different data set. Overall, CNNs can be very effective for movie poster recognition tasks, but the performance of the model depends on the quality and size of the training data set, as well as the choice of CNN architecture and other modeling techniques used.

I. Motivation

The motivation behind this project is to develop an automated and efficient method for classifying movies into different genres. Movie posters are designed to convey information about the movie and attract an audience. One important piece of information that movie posters convey is genre of the movie. If a movie's genre can be predicted by the movie's poster, then using a movie poster as data may be helpful in other neural network processes like recommendation algorithms.

Using movie posters for genre classification has practical applications in the movie industry. Since posters play a large role in the marketing efforts of movie studios, accurately predicting the genre of a movie based on its poster could be an essential tool when creating marketing materials for film promotion. Studios can make sure that the posters they are producing are in-line with the genre of film they are making. This can help studios have more targeted marketing campaigns and attract audiences who are interested in that genre. Further research can be done to advise movie studios to use certain colors or patterns when creating posters.

Traditional approaches to analyzing movie posters typically involve manual labeling or subjective interpretation. This can be time consuming and inaccurate. By using CNN multi-label classification, movie studios can efficiently analyze large numbers of movie posters and classify them into multiple genres with a high degree of accuracy. Also, by analyzing the visual elements of a poster, such as the color palette, typography, and imagery, CNN multi-label classification can identify patterns that are associated with particular genres or themes. The results can help movie studios predict how audiences will perceive the film the studio is marketing.

II. Data sets

The data set is from a website called Kaggle [1], a platform for data science and machine learning competitions where companies or individuals can host challenges. The data set contains information on movie posters and their respective genres. Kaggle provides two files: an image data file and a movie CSV file. The posters themselves are high-quality images in JPEG format, with varying sizes and aspect ratios. The data set also includes a CSV file containing the image filenames marked with a movie ID and their corresponding genre labels. The original data set contains around 40,000 movie posters, each with corresponding genre labels. The data set is intended to be used for machine learning tasks, such as image classification or genre prediction based on movie posters.

The data set includes the following features:

1) imdbId: a numeric string representing the Id associated with the film pulled from the film website http://www.imdb.com
2) imdb Link: a URL to the associated film's IMDB page
3) Title: the title of the movie and the year of its release
4) IMDB Score: the audience rating from http://www.imdb.com
5) Genre: a string representing the genre(s) of the movie, with multiple genres separated by a pipe (|) symbol
6) Poster_Path: a string representing the file path of the movie poster image

There are 28 unique movie genres in the original data set: Action, Adult, Adventure, Animation, Biography, Comedy, Crime, Documentary, Drama, Family, Fantasy, Film-Noir, Game-Show, History, Horror, Music, Musical, Mystery, News, Reality-TV, Romance, Sci-F, Short, Sport, Talk-Show, Thriller, War, Western.

The genre labels were assigned based on data available on IMDb and TMDb websites and may not accurately represent the actual genres of the movies. Additionally, some movie posters may contain multiple genres, while others may not have any genre information available. Each movie had between one to three different unique genres.

III. Related Work

The article "Analysis of top box office film poster marketing schemes based on data mining and deep learning in the context of film marketing" [2] discusses the use of data mining and deep learning techniques to analyze the effectiveness of film poster marketing strategies for top box office films. The authors collected data on the top 200 box office films from 2015-2019, including film posters, box office revenue, and critical ratings. They then used data mining and deep learning techniques to analyze the relationships between film poster features, such as color, font, and image content, and box office revenue and critical ratings. For example, they may find that posters that generate a lot of social media buzz tend to be those that are visually striking or that feature controversial or provocative imagery. The article highlights the potential of data mining and deep learning techniques to provide insights into the effectiveness of film marketing strategies and inform decision-making in the film marketing industry.

The article "Binary cross entropy with deep learning technique for Image classification" [3] discusses the benefits of using binary cross entropy as a loss function for image classification. This article specifically discusses the benefits of using this loss function in CNNs. The article uses flower classification as an example case and achieves 94% accuracy when using this loss function in conjunction with a softmax classifier. To use binary cross-entropy for multi-label classification, each label is considered as a separate binary classification task, where the goal is to predict whether the label is 0 or 1. The binary cross-entropy loss function measures the difference between the predicted probability distribution and the true probability distribution for each label. By using a binary cross-entropy loss function for each label, the model can learn to predict each label independently of the others. This approach is effective in cases where labels are not mutually exclusive and can co-occur.

IV. Technical Approach

CNN models involve several technical steps, including data preprocessing, model architecture design, and model training.

A. Data Preprocessing

Data preprocessing involves preparing the data for use in the model. This process can be categorized into three parts: data cleaning, data segmentation, and data augmentation.

After downloading the data from Kaggle, the data has to be cleaned before it can be used in CNN models. The initial data set contained a CSV with 40,000 rows which represent 40,000 different movies. These 40,000 movies encompassed 28 different movie genres, with each movie potentially being classified with multiple genres. The first step to data preprocessing is cleaning the CSV of extraneous information by removing duplicates and rows that do not have genre information associated with the film. The CSV was then randomly sampled for 6,000 movies.

A Python script then iterated through the CSV, downloaded any URL that had an associated poster linked, and downloaded the images into a folder in Google Drive. This was accomplished through the use of a python package called $urllib$ which allows the user to manipulate URLs in a python environment. The function in the package is called $request$, which opens and reads in the URL. The script then downloads these images as a JPEG file. An exception block was added to the script to simply ignore problematic URLs and go to the next image if a URL does not download properly. Through this process, the the final CSV contained 5,384 films since 616 of the 6,000 films sample did not have working poster URLs associated with them.

Due to a large number of genres and data, a subset of the data set was used to ensure the CNN models would be able to generate feasible results. The top five genres, or genres that corresponded with the most amount of movies, were selected by grouping by the unique movie genres. These genres were Drama, Comedy, Romance, Action, and Crime. With this new subset, the data set was split with 80% used for training and 20% was used for testing. The training set is used to train the CNN model, while the testing set is used to evaluate its performance.

To ensure that each movie poster was the same structure, data augmentation such as image resizing and normalization was done on each poster to ensure that all images in the data set have the same dimensions. Normalization involves scaling the pixel values to a common range, usually between 0 and 1. In addition, the genre labels, which were initially strings separated by a pipe symbol, were changed into a binary list. The binary list was the length of the number of classes, with each index corresponding to a distinct movie genre. A 1 would indicate that the movie is that genre and a 0 indicates the movie is not that genre. Each movie now has a corresponding list that will be used as the input to the CNN model.

B. Model Architecture Design

The architecture of a CNN model consists of a series of convolutional layers, pooling layers, and fully connected layers. Convolutional layers apply filters to the input image to extract features. Pooling layers reduce the spatial dimensions of the feature maps. Fully connected layers use the extracted features to make predictions. The number of layers, the size of the filters, and the number of neurons in the fully connected layers are all important design decisions that can affect the performance of the model. To observe how CNN models produce multi-label classification results, ResNet and DenseNet models were used since they are two popular deep-learning architectures that have been widely used for multi-label classification tasks. In order to compare the results, a simple custom CNN model with one drop-out layer was

used as a baseline. The four models used for experimentation are ResNet50, ResNet18, DenseNet201, and a custom CNN model.

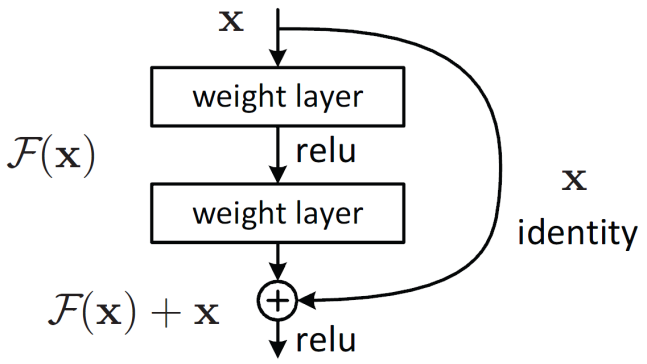

Fig. 1. Resnet Architecture. Source: [4]

ResNet is a deep neural network architecture that uses residual connections to improve the performance of the model. Residual connections allow the network to learn residual functions, which can help to overcome the vanishing gradient problem that can occur in very deep networks. ResNet models are typically used as feature extractors, where the last few layers of the network are removed, and the extracted features are fed into a classifier to make predictions. To use ResNet for multi-label classification, the output layer of the ResNet model is replaced with a sigmoid activation function instead of a softmax function. This allows the model to output probabilities for each label independently, rather than forcing the probabilities to sum to 1. The binary cross-entropy loss function is used to optimize the model for multi-label classification.

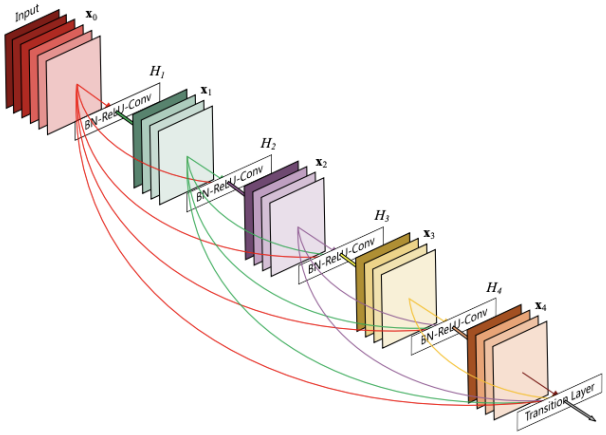

Fig. 2. Densenet Architecture. Source: [5]

DenseNet is another deep neural network architecture but uses dense connections between layers to improve the performance of the model. Dense connections allow each layer to receive input from all previous layers, which can help to improve the flow of information through the network and reduce the number of parameters required. DenseNet models are typically used as feature extractors, where the last few layers of the network are removed, and the extracted features are fed into a classifier to make predictions. To use DenseNet for multi-label classification, the output layer of the DenseNet model is replaced with a sigmoid activation function, and the binary cross-entropy loss function is used to optimize the model for multi-label classification. The number of output neurons in the last layer of the network should be set to the number of labels in the data set.

The custom CNN, which design is based on the VGG model, is shown below:

Fig. 3. Custom Neural Network

The model contains 3 convolutional layers and 3 fully connected layers. The difference between VGG and similarly simple CNNS is its use of very small convolutional filters (3x3) early in the networks. This is done for two mains reasons: to decrease the number of parameters and add more non-linearity through more ReLU layers. Multiple smaller convolutions can have the same receptive field of a larger filter (e.g. two 3x3 have an effective receptive field as a 5x5) However, smaller filters allow for an increased use of activations through this receptive field, allowing for more nonlinearity to provide a more discriminative network [6]. Multiple smaller filters also use far less parameters than their single filter counterpart. For example, our three 3x3 layers have 81% less parameters than the identical effective receptive field of a single 7x7 layer. Pooling layers are used to reduce the spatial dimensions of the feature maps produced by the convolutional layers. This model uses max pooling, which takes the maximum value from each sub-region of the feature map. Dropout layers are also used to prevent overfitting in the model. Dropout randomly drops out a fraction of the neurons in the network during training, which helps to reduce overfitting and improve the generalization of the model.

C. Model Training

The loss functions used for multi-label classification is Binary Cross Entropy. This function measures the difference between the predicted probability distribution and the true probability distribution for binary classification tasks. The true probability distribution is represented as a binary vector, where the value of the vector is 1 for the positive class and 0 for the negative class. The gradients of the loss function with respect to the model parameters are computed using back-propagation, and the model parameters are updated to minimize the loss function.

Multi-label classification models are susceptible to overfitting, which can lead to poor performance on test data. To mitigate this, regularization techniques are commonly used.

Regularization is a technique that adds a penalty term to the loss function to discourage complex models that fit the training data too closely. The CNN models run in the experiments use dropout regularization. Dropout is a technique where random nodes or connections in the neural network are dropped out during training, reducing the reliance on any single node or connection. This can prevent overfitting by encouraging the network to learn more robust and generalized features.

Regarding optimization, the choice of optimizer depends on the specific requirements of the problem and the characteristics of the data. For the experimental CNN models, the Adam optimizer was used due to the adaptive learning rates ability to update the model parameters. Adam is computationally efficient and performs well on a wide range of problems.

V. Experiments

Experiments were run on a set of CNN models and the results were analyzed based on predetermined evaluation metrics.

A. Evaluation Metrics

Overall precision, overall recall, and overall F1 score are evaluation metrics used for multi-label classification tasks. These metrics evaluate the model's performance across multiple labels and provide a single metric to compare different models. This makes it easy to compare different models and choose the best-performing one. Descriptions of the metrics are as follows:

1) **Overall precision (OP)**: Measure of the proportion of correctly predicted positive instances out of all predicted positive instances. It can be calculated by dividing the number of correctly predicted positive instances by the sum of instances predicted as positive but are actually negative.
2) **Overall recall (OR)**: Measure of the proportion of correctly predicted positive instances out of all actual positive instances. It can be calculated by dividing the number of true positives by the sum of instances predicted as negative but are actually positive.
3) **Overall F1 score (OF1)**: Harmonic mean of precision and recall, and it provides a balanced evaluation of both metrics. It can be calculated as

$$\frac{2*(precision*recall)}{(precision+recall)}$$

The reason overall F1 score provides a good method to compare different models is the trade-off between precision and recall. This provides a balanced evaluation of the model's performance and is particularly useful in multi-label classification tasks where the model has to make multiple predictions for each input. In addition, the method is easy to interpret and widely used in the deep learning community.

B. Analysis

Four different CNN models were run on the multi-label movie poster data set. Each model was run using 5 epochs and 10 epochs. The learning rate was tested with both 0.001 and 0.0001 to determine which rate would provide the best results. Each iteration of the learning rate was run with each iteration of the epoch count. The batch size was initially set as 32 but due to the size of the data set, increased to 64 in order to see noticeable changes in the evaluation metrics during the train and test loops.

After viewing the change in results with the adjustments to the hyperparameters, the finalized values used were 10 epochs, with a learning rate of 0.0001. Running the models with more epochs allowed a clear view of when model metrics started to plateau. A smaller learning rate helps a model converge to the optimal solution more accurately and quickly, especially in complex optimization problems. By reducing the step size of the weight updates during training, a smaller learning rate can prevent the model from overshooting the minimum of the loss function. However, increases the training time and requires more computational resources.

During each batch during the training loop, an OP, OR, and OF1 score was calculated. The testing loop takes a subset of the data and calculates how close the model fits with new data. The same metrics were calculated to compare with the training loop results. By having the same evaluation metrics for all the models, issues like overfitting can be discovered and trends between the different data can be observed.

C. Results

After running all four models through both a train and test loop, these were the results:

Model	OP	OR	OF1
ResNet18	0.6605	0.4844	0.5577
ResNet50	0.5760	0.5760	0.4796
DenseNet201	**0.6984**	**0.5789**	**0.6331**
Custom CNN	0.6533	0.2740	0.3852

A higher F1 score indicates the better overall performance of the model. This means the model is able to identify most of the positive samples in the data set and indicates that the model makes very few false positive predictions. The DenseNet performed the best based on this evaluation metric. The custom CNN was used as a benchmark to determine whether any of the other models were overfitting. When observing the training OF1 scores per each epoch and comparing it with the testing OF1 scores, the difference was negligible.

While the F1 score showed a clear difference, the loss for each model changed differently. The DenseNet started with good results but the loss didn't decrease as much as the other models. On the other hand, both ResNet functions saw significant changes over the 10 epochs. The use of preset weights is mostly the reason for this delta. This could cause the models to start at different parts of the learning process.

VI. Conclusions

In conclusion, CNNs are a powerful deep-learning architecture for multi-label image classification. There are a few takeaways from the CNN experimentation. CNNs are effective in capturing complex features from the input image that

are important for classification. In multi-label classification, this allows the model to make multiple predictions for each input, improving the accuracy of the classification. Transfer learning, the process of using a pre-trained model as a starting point, can significantly improve the performance of CNNs in multi-label classification tasks, even with limited data. Overall precision, overall recall, and overall F1 score are commonly used evaluation metrics for multi-label classification tasks in deep learning, including CNNs. These metrics provide a comprehensive and balanced evaluation of the model's performance, allowing for easy comparison with other models and interpretation.

The CNN models used in this experiment varied in their classification capacity with the Custom CNN performing the worst, yielding an F1 score of 0.3852, and the DenseNet performing the best, yielding an F1 score of 0.6331. These result are expected since the DenseNet model is a much more complex, pre-trained model consisting of 201 layers when compared to the custom model which only has 10 layers. While no model had a very high F1 score, all the models used were still significantly better at predicting movie genre based on the poster than compared to random chance.

Overall, using CNN for multi-label classification is a powerful approach that can produce accurate results when used effectively. By leveraging the strengths of CNNs and carefully selecting evaluation metrics, it is possible to build high-performing multi-label classification models.

VII. Resources

All code for this project can be found on our GitHub at the following address: https://github.com/theodore-thormann/DS6050_FinalProject

References

[1] N. Neha, "Customer Segmentation using k-means Clustering," Predicting the Genre of the movie by analyzing its poster. Retrieved April 22, 2023, from https://www.kaggle.com/datasets/neha1703/movie-genre-from-its-poster.

[2] S. Yang, "Analysis of top box office film poster marketing scheme based on data mining and deep learning in the context of film marketing," PLoS One, vol. 18, no. 1, e0280848, Jan. 26, 2023. Retrieved April 22, 2023 from https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0280848

[3] U. Ruby and V. Yendapalli, "Binary cross entropy with deep learning technique for image classification," Int. J. Adv. Trends Comput. Sci. Eng, vol. 9, no. 10, pp. 2876-2882, 2020. Retrieved April 22, 2023 from https://doi.org/10.30534/ijatcse/2020/175942020

[4] "Residual Network (ResNet) Architecture," Neurohive, [Online]. Retrieved April 27, 2023 from https://neurohive.io/wp-content/uploads/2019/01/resnet-e1548261477164.png.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks," arXiv:1608.06993 [cs.CV], Aug. 2016. [Online]. Retrieved April 27, 2023 from https://arxiv.org/abs/1608.06993

[6] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 3rd International Conference on Learning Representations (ICLR 2015), Computational and Biological Learning Society, 2015, pp. 1-14. Retrieved April 27, 2023 from https://arxiv.org/abs/1409.1556