Movie Genre Prediction from Posters

Manish Kolla
Machine Learning
(CS 4850)
Department of Computer Science
Atlanta, GA, United States
mkolla1@student.gsu.edu

Jainam Shah
Machine Learning
(CS 4850)
Department of Computer Science
Atlanta, GA, United States
jshah23@student.gsu.edu

Lipika Arya
Machine Learning
(CS 4850)
Department of Computer Science
Atlanta, GA, United States
larya1@student.gsu.edu

***Abstract*— This paper proposes a Convolutional Neural Network (CNN) based method for automatic movie genre classification from poster images. The proposed CNN architecture utilizes a neural network trained on a dataset of movie posters. The paper details the CNN design, training process, and data pre-processing techniques employed. Data pre-processing includes one-hot encoding genre labels, handling missing values, addressing data imbalance, and image re-sizing/normalization. The performance of the proposed CNN is evaluated and compared against established models like LeNet, AlexNet, VGG variants, ResNet-50, Logistic Regression, and Random Forest.**

Keywords: *Prediction, Random Forest, neural networks, lenet, alexnet, vgg-16,vgg-19, resnet, logictTensorFlowion, adam, sgd optimizers, tensorflow, keras*

I. Introduction

The goal of this project is to close the perception gap between audience perception and movie poster design. We give poster artists a useful tool by creating an automated movie genre classification system based on poster imagery. They can use this tool to customise their designs so they can appropriately express the genre of the film and effectively attract the attention of viewers.

In this article, we offer a method for classifying movie posters into genres using Convolutional Neural Networks (CNNs). We use a large collection of movie posters to train our multi-layered CNN architecture. We actively experiment with multiple training parameters and numerous changes to the original network design in order to get optimal performance. We are able to extract both high-level and low-level information from the photos thanks to our methodical technique, which results in extremely precise genre classification.

The potential for this system to be implemented successfully is to greatly enhance movie recommendation systems. It is possible to make recommendations that are more user-focused and targeted by utilizing genre classification based on poster analysis. Furthermore, our research's conclusions might offer insightful advice to poster designers, empowering them to create images that successfully convey the genre of the film and appeal to the intended audience.

GitHub Link:
https://github.com/manishkolla/Movie-Genre-Recognition-from-Posters

Related Works:
One such related work similar to this project idea we found is this research paper [1] which we have decided to take up as a reference and develop an application-based project, implementing the originally mentioned models in the research paper and also learning the original model architecture and their tuning of the hyperparameters and checking for possible factors that might affect the accuracy. Also, in the mentioned paper above researchers proposed an alternative model that can perform with higher accuracy compared to other baseline models mentioned by them in the paper. This model was trained on higher dimensions of parameters and more training data to attain accuracy. However, due to our technological limitations, we could not achieve the accuracies mentioned in the paper. The paper also illustrated the graphs for various models and their accuracies along with various evaluation metrics.

Work Distribution:

Manish Kolla: Complete implementation of Lenet, Alexnet, and proposed model including its experimental analysis and evaluations.

Jainam Shah: Complete implementation of VGG -16, VGG-19, and Logistic Regression including its experimental analysis and evaluations.

Lipika Arya: Complete implementation of Resnet -50, and Random Forest including its experimental analysis and evaluations.

Life Cycle of this Project
Throughout our project we have worked on following the CRISP-DM life cycle using the following phases

1. Business Understanding (Understanding the final deliverables)
2. Data Understanding (Data Availability, Features understanding and selection)
3. Data Processing (handling null values, molding the data, and merging them)
4. Modeling (Implementation of several Deep Learning and ML Algorithms that can be used with this data)
5. Evaluation (Measuring the loss, accuracy, ROC, and Confusion Matrix)
6. Deployment (Implementing a prediction function)

Business Understanding:
Inaccurate movie posters can mislead viewers and hurt box office sales. This project aims to develop a more accurate movie genre classification system using a custom CNN. This will empower poster designers to create targeted

visuals and improve movie recommendation systems, ultimately boosting audience satisfaction and potentially increasing revenue.

Data Understanding:

Data Source: The data is taken from a source like IMDB, containing movie information and corresponding poster links from Kaggle Open-Source Data. Due to technological limitations and memory limitations, we could not work with multiple datasets as this dataset which we chose has up to 30,000 different movies and related posters and genres.
Data Description:

The dataset includes various attributes about movies: IMDB ID, link, title, score, genre(s), and a downloadable poster link. Each movie poster can be associated with at least one and up to three genres. The label data, which is the target variable for our model, consists of the genre(s) linked to each movie poster. These genres encompass various categories like action, romance, comedy, drama, and more.

Quality Check: Not all the poster links are retrievable and some of them were missing also there are a very minimal number of null values in the genres columns, so imputation was not necessary for this dataset.

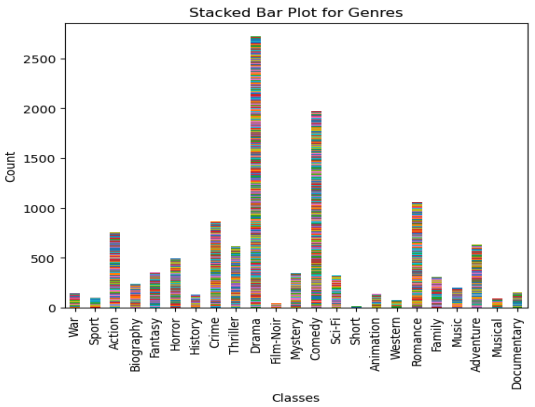
Data Preprocessing:

Drop the features:
Initially the dataset has 6 main features as mentioned above as we will not be using all the features for training some of them were removed such as IMDB Link, IMDB Score, and Title. After filtering them out we were only left with 3 main features namely Poster, Genre, and IMDB ID.

Dropping the missing values:
Two features have a very minimal number of null values which are poster and genre and as the percentage of null values is very minimal, we did not choose to impute them with manual imputation instead we removed them as the effect is negligible.

One-Hot Encoding:
It splits genres listed as strings into individual genres, identifies all unique genres, and creates a new binary column for each unique genre. This one-hot encoding assigns a 1 to a genre if it's present for a movie and a 0 otherwise, transforming the data into a format suitable for machine learning models.

Removing the Genre below the mean count:
We visualized all the genres using the stacked bar plot to evaluate and get more understanding of the value counts of each genre as we have around 23 different genres. Below is the stacked bar plot visualization.



Based on this plot we concluded that there are many genres with huge imbalances which might lead to the majority of the predictions to predict the majority class label which in this case is the Drama followed by Comedy and this might cause the minority labels to get compromised. Due to this, we have decided to remove the genres that are below the mean number of value counts. In the previous visualization genres such as short, western, and film-noir were dropped to focus more on balanced data. After filtering them out we are left with 7 genres as visualized below.

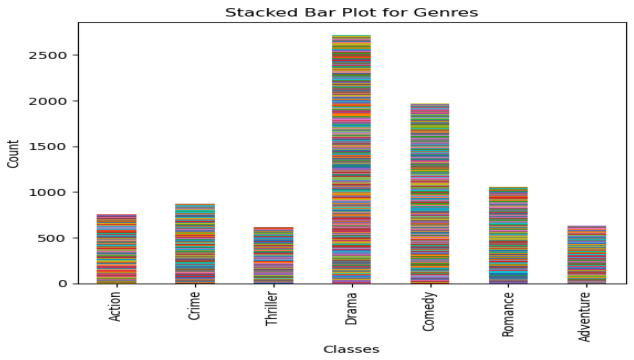


Image Processing:

Our image preprocessing methodology involves several key steps to ensure easier and light model training.

Image Loading: We implemented a multi way image reading using URLlib from python which can read images from urls. If this fails, then we are using the Python Imaging Library (PIL) for local file handling using a try except case. This flexibility ensures the model to handle different image locations without the necessity of changing everytime.

Resizing and Grayscale Conversion: All images are resized to a standard size of 224x224 pixels. This standardization simplifies image processing for the convolutional neural network (CNN) by ensuring consistent input dimensions throughout all the images. Additionally, images are converted to grayscale format for easier training

of the data as it does not need higher memory space in storing and training when in comparison to RGB and HSV. While discarding color information, grayscale conversion reduces data complexity while preserving essential details relevant to genre classification to a certain extent.

Array Conversion and Normalization: Following conversion to grayscale, images are transformed into NumPy arrays. Pixel values within the image array are then normalized to a range between 0 and 1 for easier calculation and training of the model and can also help in faster convergence. This normalization step facilitates training by ensuring all features are on a similar scale.

Batch Dimension Addition: Finally, a batch dimension is added to the preprocessed image array. This additional dimension enables the model to process multiple images efficiently in a single batch during training.

Modeling:
Throughout this project, we have used several Neural network models and two machine learning algorithms which include:

1. LeNet
2. Alex Net
3. Proposed Model (Reference to paper)
4. VGG-16
5. VGG-19
6. Resnet- 50
7. Logistic Regression
8. Random Forest

For all the above models similar preprocessing has been done to maintain the consistency of the training balance the parameters and compare accuracies among each other. All the above models have been extensively experimented with and evaluated with varying hyperparameters which include:

1. Change in Epochs.
2. Change in Activation Functions.
3. Change in batch size.
4. Change in optimizers.
5. Addition of new hidden layers with varying neurons
6. Removal of some hidden layers.
7. Changing in learning rates.

LeNet Implementation and Performance Analysis

This section details the development and evaluation of a LeNet-based convolutional neural network (CNN) for genre classification.

Model Architecture: We implemented a custom LeNet architecture with the following characteristics and parameters:

Convolutional Layers: Two convolutional layers were used, utilizing filter sizes of 6 and 16, respectively. Padding strategies includes "same" and "valid" to control output dimensions and balance the image while pooling.

Activation Function: Sigmoid activation was applied for non-linearity within the convolutional layers as mentioned in the original documentation of Le Net.

Pooling: Max pooling with a pool size of 2 and 2 strides were used to retrieve the essential features of the image.

Fully Connected Layers: Two fully connected layers were utilized, containing 120 and 84 neurons each. The final output layer possessed a neuron count corresponding to the number of genres after the filtering.

Baseline Performance: The initial implementation achieved a testing accuracy of 24%. Analysis showed that class imbalance, with "drama" dominating, followed by "comedy" and "romance."

Hyperparameter Tuning: To improve performance, a series of experiments were conducted which includes:

Epoch Variation: The number of training epochs was explored, ranging from 5 to 50.

Batch Size Adjustment: Batch sizes were varied between 10 and 64 to assess their impact on accuracy.

Hidden Layer Addition: An additional hidden layer with 42 neurons was incorporated before the output layer.

Optimizer Selection: Two distinct optimizers were been experimented with, Adam and SGD, and were compared. Adam emerged as the superior choice due to its higher accuracy over SGD at most of the situation.

Loss Function: Categorical cross-entropy was used as the loss function, being the best among for multi-class classification tasks.

Results and Discussion:

Three LeNet variations were evaluated:

1. **Original Architecture**: This baseline model achieved the previously mentioned 42.9% accuracy.
2. **Additional Hidden Layer**: The addition of an extra hidden layer yielded average improvement.

Models	(5/15, 16/32, Adam/SGD)	(5/15, 64, Adam/SGD)	(5, 64, Adam/SGD)	(15, 64, SGD)
Model- 1	22.7	22.7	22.7	22.7
Model- 2	22.7	11.16	22.7	22.7
Model- 3	(5/15,16,Adam/SGD): 22.7 (15,32,SGD): 11.26	(15, 64, Adam): 22.7	(5,64,SGD):11.16; (5, 64, Adam): 22.7	14.7

Table -1

3. **Reduced Final Layer Neurons**: Replacing the original 84-neuron final layer with a 40-neuron layer resulted in the most significant performance increase, reaching an accuracy of 64.16% for once.

Model Conclusion:

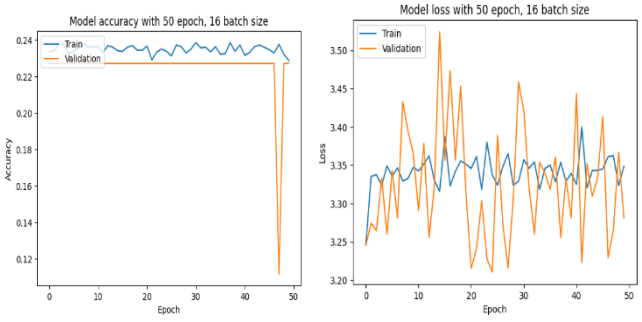
This model concluded with several experiments stating the different accuracies with varying model parameters played a vital role. By implementing a reduced final layer and employing the Adam optimizer, a significant accuracy improvement was achieved on a dataset of 3148 images containing seven unique genres.

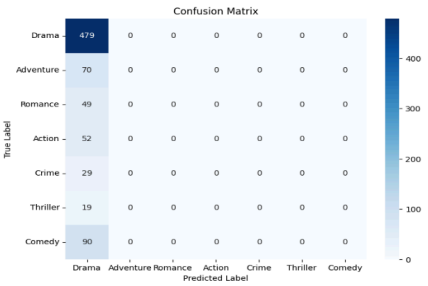
Complete detailed experimental results and their training/validation accuracies are mentioned below in Table 1. The table follows a legend for easier navigation of accuracies with table legend (Epoch, Batch size, optimizer).

Model 1,2,3 reciprocates the Lenet model variants as mentioned above.

Evaluation for Le Net:

For the previously mentioned Let Net, here are the training and testing/validation accuracies and loss for 50 epochs and batch size 16 using the Adam optimizer:





As we can observe as the epoch increases, the accuracy stays the same and is consistent due to early convergence. Also, the loss has been fluctuating a lot without any linear or exponential growth. Out of curiosity, we tried to visualize the confusion matrix to see how the truth and prediction labels are, and out of astonishment, we found that all the labels are first being predicted as Drama due to their superior count of labels. We have also tried to perform analysis by removing the column Drama but it did not change the accuracies, however, the predictions changed to

Comedy instead of Drama this time. So, we did not see any advantage in removing the column drama due to which we did not remove the column. One interesting finding is, that at epoch numbers 45-50, there is a huge drop in the accuracy.

Compared to the modified models, the original architecture model has stayed consistent regarding the number of parameter changes we made it persisted with the same accuracy which led us to the doors of new learning for the reasons to explore and it turns out the possible reasons are: underfitting, insufficient training parameters/weights, and model architecture.

Alex Net Implementation and Performance Analysis

The Alex Net architecture, a convolutional neural network (CNN) for image classification, is implemented in this work. The architecture leverages multiple convolutional layers followed by pooling layers to extract increasingly complex features from the input image. Here's a breakdown of the key layer properties:

Parameters	(5/15,16)	(5,64)	(15, 64)	(25,64)
Adam	31.8	19.4	38.4	38.4
SGD	31.8	38.4	19.4	38.4

Table -2

Number of filters: The number of filters in each convolutional layer progressively increases (96, 256, 384, 384, 256). This allows the network to learn a wider range of features from simple edges in the initial layers to more intricate object parts in the later layers.

Kernel size: A consistent 3x3 kernel size is used throughout the convolutional layers. This kernel size is often a good choice for capturing local spatial information in images.

Activation functions: ReLU (Rectified Linear Unit) activation is used after each convolutional layer. ReLU introduces non-linearity, allowing the network to learn more complex relationships between features.

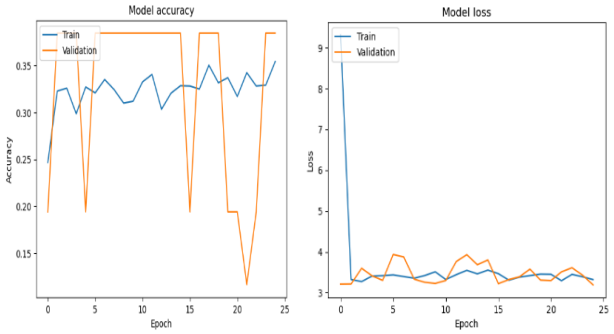
Pooling layers: Max pooling layers (pool size 3x3, strides of 2) areused. Pooling reduces the dimensionality of the data, improves computational efficiency, and introduces some invariance to small shifts in the object's position within the image.

Fully connected layers: The extracted features are flattened and passed into two fully connected layers with 4096 neurons each. These layers perform classification on the learned features. Dropout (0.5 probability) is applied after each fully connected layer to prevent overfitting, a common challenge in machine learning where the model performs well on training data but poorly on unseen data.

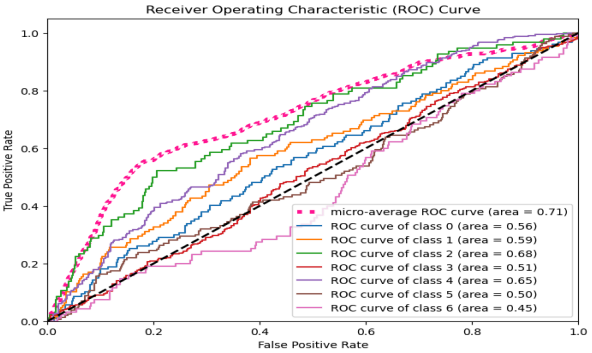
Finally, a softmax activation function is used in the output layer. Softmax outputs a probability distribution over this model. This model by far used the most number of neurons and also used the dropout function. The model accuracies and the experimental analysis can be found above in Table 2.

For easier navigation, a legend is provided.
Table legend (Epoch, Batch size) with a learning rate of 0.001 and 0.0001.

Evaluation of Alex Net:
Due to the higher number of neurons in the hidden layer this model took a comparatively huge amount of time for the experimental analysis compared to the previous models. Similar evaluation analysis has also been done on this model and here are the visualizations we encountered. For 25 epochs and batch size 64 with Adam optimizer. One of the interesting findings of this model is, that the validation accuracy turned out to be higher than the training accuracy which we researched and found some of the possible reasons as model complexity due to overfitting, insufficient training size, and shuffled splitting of training and testing data.



We have also visualized the AUC-ROC curve for this model to see the prediction if they are TPR or FPR and most of the labels tend be to higher than the TPR, which might be possibly due to the majority class label.



Proposed Model and Experimental Analysis:

This is the proposed model proposed by the researchers in the paper. This model has been designed to achieve a higher accuracy compared to all other previous models mentioned. Researchers designed this model using the following parameters:

Number of filters: This defines the dimensionality of the filters used in the convolution layer. The model increases the number of filters (16,32,64,128) as it goes deeper, allowing it to capture a wider range of features..

Kernel size: This hyperparameter specifies the dimensions of the filter used for convolution. In this model, all convolutional layers utilize a 5x5 kernel size.

Activation function: The activation function introduces a non-linearity function. Here, the ReLU (Rectified Linear Unit) activation function is used in all layers.

The padding argument within each convolutional layer is set to "same". This ensures the output feature maps retain the same spatial dimensions (height and width) as the corresponding input.

Pooling Layers: Following each convolutional layer, a max pooling layer is applied. These layers perform down-sampling operations, reducing the resolution of the feature maps. This technique helps to control model complexity and overcome overfitting during the training

process. The model utilizes max pooling with a pool size of 2 and strides of 2.

Fully Connected Layers: These layers perform linear transformations on the flattened feature vector, learning decision boundaries to separate the data into distinct classes of genres. Each fully connected layer is defined by two key hyperparameters:

Number of neurons: This value represents the number of neurons within the layer. The first two fully connected layers each possess 128 neurons. The final layer, responsible for the final class probabilities, has several neurons equivalent to the specified number of unique genres after filtering.

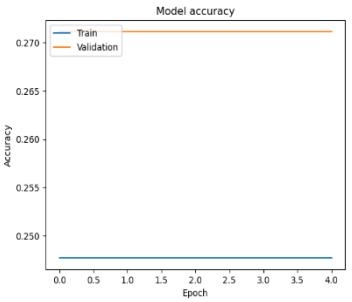
Activation function: The activation function applied in each fully connected layer influences how the layer transforms its inputs. While the first two layers utilize the sigmoid activation function, the final layer employs the softmax activation. The softmax function ensures the output probability sums to one and represents probabilities for each predicted class.

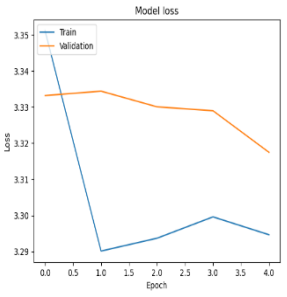
Accuracies with varying changes in parameters have been recorded and here is the legend for the table readings:
Table legend (Epoch, Batch size, optimizer)

Model Parameters	Accuracy
(5/15, 16, Adam/SGD)	27.11
(5, 64, Adam/SGD)	27.11
(15, 64, SGD)	13.11

Evaluation of Proposed Model

Similar to the results in the Alex Net, we encountered similar results in the proposed model where the testing accuracy was higher than the training accuracy and the possible reasons might be similar. We also tried to experiment with increasing the parameters but we have been consistently getting memory errors due to declarations of numpy arrays with such huge data. Based on the limited parameters trained here are the accuracy change and loss to the changing epoch sizes. We have also visualized this using a confusion matrix and it turns out the same scenario is happening is similar to the lenet model where it predicts the drama/comedy with a higher probability.





VGG-16 Implementation and Performance Analysis

VGG16

Convolutional Layers:

Number of Filters: In the VGG16 model, every convolutional layer includes a unique set of filters, which function as specialized detectors for various picture properties. The number of filters is gradually increased by the model. There are 64 filters in Conv-1 Layer, 128 filters in Conv-2, 256 filters in Conv-3, and 512 filters in Conv-4 and Conv-5.

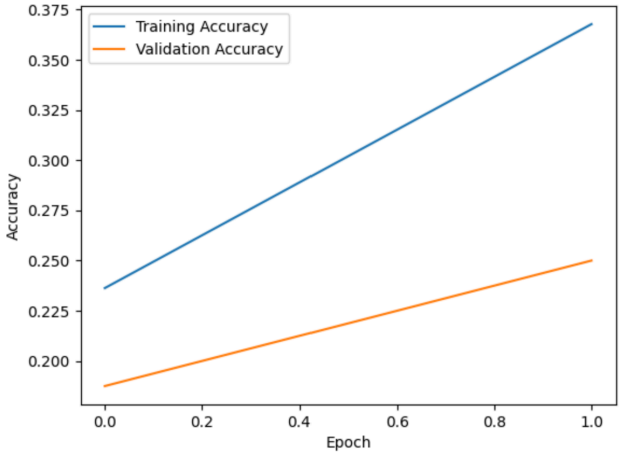
Kernel Size: The VGG16 design also predefines the kernel size, which establishes the region that the filter operates over. Every convolutional layer in this model uses a 3x3 kernel size.

Activation Function: Rectified Linear Unit, or ReLU, is the activation function that is typically employed in convolutional layers. The model gains non-linearity from

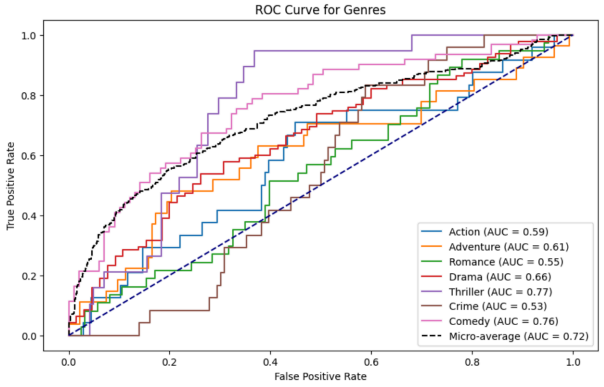
this function, which enables it to recognize complex patterns in the data.

On top of the VGG16 base model, a custom model is constructed with fully connected layers. In the provided code, a Dense layer with 512 units and a ReLU activation function is added first. Then, another Dense layer with 7 units is added, representing the number of output classes. This final layer uses the softmax activation function, which converts the output into probabilities for each class. These fully connected layers play a crucial role in extracting higher-level features from the convolutional layers and making predictions based on those features.

Here is the accuracy of the VGG-16 model:

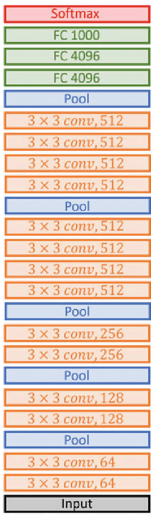


Here is the ROC curve of genres for the VGG-16 model:




```
ROC AUC score for Action: 0.5860906862745098
ROC AUC score for Adventure: 0.609857978279031
ROC AUC score for Romance: 0.5460338387167656
ROC AUC score for Drama: 0.6563562753036437
ROC AUC score for Thriller: 0.7678238148562897
ROC AUC score for Crime: 0.5346200980392157
ROC AUC score for Comedy: 0.761384335154827
Micro-average ROC AUC score: 0.7222540584177922
```

VGG-19 Implementation and Performance Analysis



VGG19

Convolutional Layers:

Number of Filters: In the VGG16 model, every convolutional layer includes a unique set of filters that act as specialized detectors for various features in the images. The number of filters is gradually increased by the model. There are 64 filters in Conv-1 Layer, 128 filters in Conv-2, 256 filters in Conv-3, and 512 filters in Conv-4 and Conv-5.

Kernel Size: In VGG19, a kernel is used by each filter to move it over the input image. The amount of the image that the filter covers at once is determined by the size of this kernel. Every convolutional layer in this model uses a 3x3 kernel size.

Activation Function: Following each convolutional layer, VGG19 applies an activation function known as ReLU (Rectified Linear Unit) in order to add complexity and learn from the data. ReLU makes sure that every layer produces a nonlinear output, which allows the images to contain complex patterns.

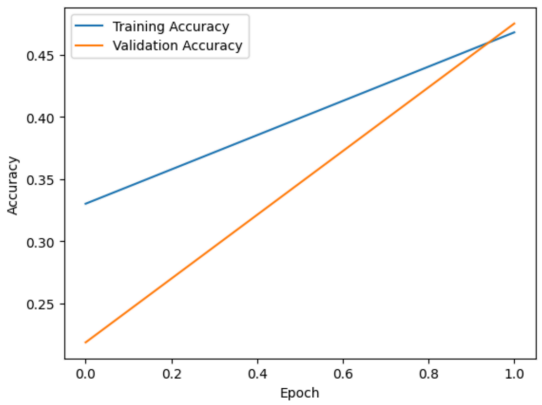
Pooling Layers:

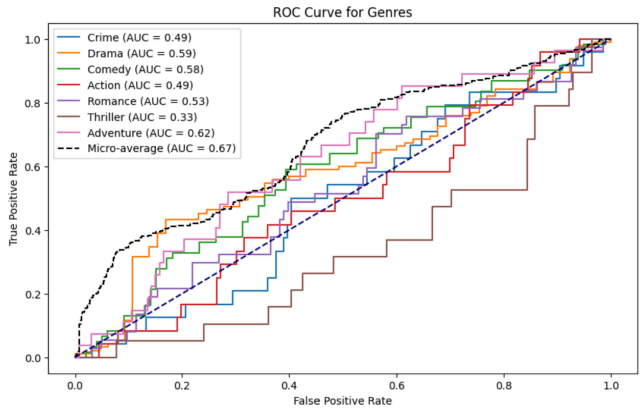
Max-pooling layers are included in VGG19 after a few convolutional layers. The spatial scale of the feature maps produced by the convolutional layers is lessened with the aid of these layers. In doing so, they efficiently downsample the data while keeping the most crucial details by choosing the largest value from each little area of the feature map. Three fully-connected layers and five max-pooling layers are present.

Fully Connected Layers:

Fully connected layers are added to the convolutional layers in the VGG19-based model. An additional Dense layer with 7 units (assuming there are 7 output classes) and a softmax activation function follows the first Dense layer with 512 units and a ReLU activation function in the code. These

layers aid in the model's comprehension of intricate correlations seen in the data and enable it to anticipate the class to which an image will belong.






```
ROC AUC score for Crime: 0.4917279411764706
ROC AUC score for Drama: 0.5863967611336032
ROC AUC score for Comedy: 0.5828779599271402
ROC AUC score for Action: 0.4892769607843137
ROC AUC score for Romance: 0.5282355526257966
ROC AUC score for Thriller: 0.3348264277715566
ROC AUC score for Adventure: 0.621275410749095
Micro-average ROC AUC score: 0.6665342095026165
```

Res Net – 50 Implementation and Performance Analysis

Model Architecture: We implemented a unique convolutional neural network (CNN) with the following features, drawing inspiration from ResNet:

Base Pre-trained Model: The main architecture for feature extraction was based on the ResNet50 pre-trained model. Weights that were trained on the ImageNet dataset were used to initialize the model.

Fine-tuning: We stopped all but the last classification layer of the pre-trained ResNet50 model. As a result, we were able to modify the model to fit our particular genre categorization assignment while maintaining the learnt features.

Additional Layers: We started with the ResNet50 basic model and added two Dense layers of 512 and 7 neurons, respectively, after flattening the 4D tensor output into a 1D vector. Seven neurons made up the final Dense layer, which matched the number of predicted genres.

Activation Function: For multi-class classification, we implemented softmax activation in the final layer after using ReLU activation for the Dense layers.

Optimizer and Loss Function: For multi-class classification problems, we used the Adam optimizer with a learning rate of 0.001 and categorical cross-entropy as the loss function.

Baseline Performance: A testing accuracy of 24% was attained in the first installation. Potential class imbalances were identified by analysis, and more optimization was required.

Hyperparameter Tuning: A number of tests were carried out in order to enhance performance.

Epoch Variation: Since the maximum was observed in the range, we investigated the number of training epochs, ranging from 1 to 50, to determine the ideal training period.

Adjusting the Batch Size: To evaluate the effect of batch sizes on training convergence and efficiency, a range of 10 to 64 was used.

Data Preprocessing: At first, we did not change the image into grayscale and we removed the ‘Drama’ genre because of the high imbalance dataset, which resulted in 40% accuracy but was eventually decreased with the increase in the epoch. Some other adjustments included changing the final dense layer to 6 neurons and the value of epoch to 2. We also made the drama genre by having equal row values of drama as there are non-drama genres, but that did not give good accuracy changes.

Results and Discussion:
Three ResNet variations were evaluated:

1. Original Architecture: This baseline configuration achieved a testing accuracy of 24% but the graph was linearly increasing. Analysis revealed potential class imbalances and the need for further optimization.

2. Additional Dense Layers: The addition of extra Dense layers or adjusting the number of neurons in existing layers yielded modest improvements in accuracy.

3. Optimizer and Learning Rate: Optimizer selection and learning rate adjustments played a crucial role in improving model performance. Adam optimizer with a learning rate of 0.001 emerged as the superior choice.

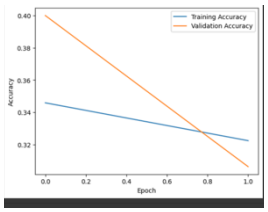


Figure 1

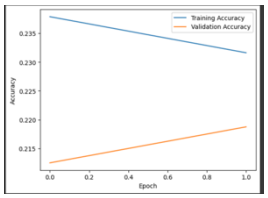


Figure 2

These figures show the Accuracy vs Epoch plot for the Resnet Model with 2 epoch values. Note: Figure 1 is without the Drama genre and Figure 2 is with the Drama Genre.

Logistic Regression

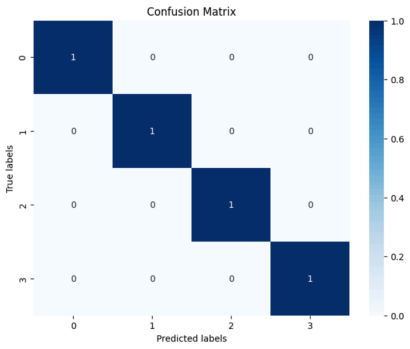
Number of Filters: Unlike convolutional layers, logistic regression does not need filters. Rather than utilizing convolutional techniques, the model makes direct use of the input features.

Kernel Size: Since convolutional techniques are not used in logistic regression, a kernel size specification is not necessary.

Activation Function: The softmax function is the activation function that is employed in the logistic regression model's dense layer. To guarantee that the output values add up to 1 for every class, this activation function transforms the raw output scores into probabilities.

Pooling Layers: Unlike convolutional neural networks, which undertake downsampling operations, logistic regression does not include pooling layers.

Fully Connected Layers: A single dense layer with softmax activation makes up a logistic regression. The code provided adds a dense layer to the model for classification, consisting of 5 units, along with softmax activation. This layer generates class probabilities by linking each input feature to each output class.



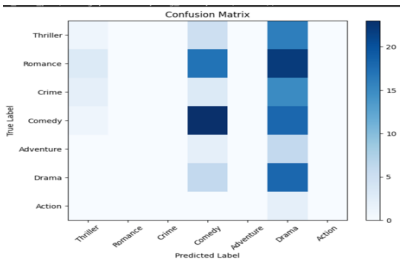
Random Forest

Preprocessing: Image resizing to 224x224 pixels. Conversion to RGB format and normalization of pixel values to [0, 1]. Flattening of poster images to create feature vectors.

Training: Splitting of the dataset into training and validation sets with a 70-30 ratio. Training of the Random Forest classifier with 200 estimators.

Model Evaluation:

Validation Accuracy: 0.19375



	precision	recall	f1-score	support
Thriller	**0.14**	**0.05**	**0.07**	**22**
Romance	**0.00**	**0.00**	**0.00**	**42**
Crime	**0.00**	**0.00**	**0.00**	**20**
Comedy	**0.41**	**0.55**	**0.47**	**42**
Adventure	**0.00**	**0.00**	**0.00**	**8**
Drama	**0.19**	**0.75**	**0.30**	**24**
Action	**0.00**	**0.00**	**0.00**	**2**
accuracy			**0.26**	**160**
macro avg	**0.11**	**0.19**	**0.12**	**160**
weighted avg	**0.16**	**0.26**	**0.18**	**160**

Discussion: The Random Forest classifier got highest accuracy of 0.19375 on the validation set. Precision, recall, and F1-score were different for different genres, with some genres showing better prediction accuracy. Some of the Challenges include the model's struggle to generalize well to a few genres and performing less accurately. We tried to find the best parameters using randomized search which showed the best estimators to use would be 443, but did not really resulted in advancement of accuracy.

Conclusion:

To conclude, this study suggests a deep neural network-based automated image-based method for categorizing different movie genres. The method makes use of both low-level and high-level characteristics taken from movie posters. To find genre possibilities, the created classifier can analyze object data and visual characteristics.

This system facilitates the production of movie posters that appeal to the intended audience and can suggest films based on related genres. When compared to the baseline reference models in the study, a few of the models in this paper have attained higher accuracy. All things considered, this technology can help moviegoers by making recommendations for new movies based on their preferences and help filmmakers create compelling marketing content.

Future Works:

With additional processing power, we can handle more datasets and train more intricate models, which improves machine learning task accuracy. Through the utilization of sophisticated technology and optimization methods, we may accelerate training periods and obtain a more profound understanding of data in numerous fields.

Work Distribution:

Manish Kolla: Complete implementation of Lenet, Alexnet, and proposed model including its experimental analysis and evaluations.

Jainam Shah: Complete implementation of VGG -16, VGG-19, and Logistic Regression including its experimental analysis and evaluations.

Lipika Arya: Complete implementation of Resnet -50, and Random Forest including its experimental analysis and evaluations.

References:

[1] N. Hossain, M. M. Ahamad, S. Aktar and M. A. Moni, "Movie Genre Classification with Deep Neural Network using Poster Images," 2021 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD), Dhaka, Bangladesh, 2021, pp. 195-199, doi: 10.1109/ICICT4SD50815.2021.9396778. keywords: {Visualization;Shape;Neural networks;Motion pictures;Feature extraction;Convolutional neural networks;Sustainable development;CNN;Feature Extraction;Multi-label classification;Deep Learning}

[2] Wikipedia contributors. (2023, December 10). LeNet. Wikipedia. https://en.wikipedia.org/wiki/LeNet

[3] Beginner's Guide to VGG16 implementation in Keras. Built In. (n.d.). https://builtin.com/machine-learning/vgg16

[4]K_02 understanding of VGG-16, VGG-19 - en. 위키독스. (n.d.). https://wikidocs.net/165427

[5] Kundu, N. (2023, January 23). *Exploring Resnet50: An in-depth look at the model architecture and code implementation*. Medium. https://medium.com/@nitishkundu1993/exploring-resnet50-an-in-depth-look-at-the-model-architecture-and-code-implementation-d8d8fa67e46f

[6] YouTube. (2021, June 29). Transfer learning using keras(resnet-50)| Complete python tutorial|. YouTube. https://www.youtube.com/watch?v=JcU72smpLJk

[7] GeeksforGeeks. (2024, March 11). Random Forest for image classification using opencv. https://www.geeksforgeeks.org/random-forest-for-image-classification-using-opencv/

[8] Saxena, S. (2023, July 18). Introduction to the architecture of Alexnet. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-the-architecture-of-alexnet/