Assignment 2 Report



Name: David Naddall

Student Email: [659586@student.inholland.nl](mailto:659586@student.inholland.nl)

# Introduction to the Dataset

## Overview of the Dataset

### Dataset Components

Movie\_Poster\_Dataset: This zip file contains multiple folders categorized by year, ranging from 1980 to 2015. Each folder is populated with JPEG images of movie posters, with the filenames corresponding to their respective IMDb IDs.

Movie\_Poster\_Metadata: This zip file comprises text files, each labelled with a specific year, containing metadata for movies released in that year. The metadata includes details such as creators, actors, genres, and release years. The dataset comprises a total of 8,873 entries across all text files.

### Dataset Format and Encoding Issues

The text files for the years 1980 and 1981 were encoded in ASCII, while files from subsequent years were in UTF-16 encoding. This inconsistency presented initial challenges in data processing.

The metadata was intended to be formatted in JSON, but numerous errors prevented it from being parsed correctly. Regular expressions (regex) were employed to correct these errors and standardize the data format.

## Data Processing and Cleaning

To manage the encoding discrepancies and facilitate data processing, all text files were first consolidated into a single large string. This string was then formatted to conform to JSON standards and converted into a unified UTF-16 format.

For ease of exploration and analysis, the JSON-formatted data was then converted into a CSV file using Python's pandas library.

The Movie Poster dataset was loaded into a DataFrame and merged with the metadata based on the IMDb ID and the title of the file of the poster. This integration was crucial for correlating posters with their respective metadata.

### Null Values

The focus was narrowed to essential fields such as IMDb ID, Genre, and Poster Path for the scope of this project. Entries with null values in these critical fields were identified and excluded from the dataset to ensure data integrity and relevance for subsequent modelling. After this 8052 entries were left in the dataframe.

### Handling multiple genres

In our dataset, movie genres were listed as strings, which is typically easy to handle. However, an issue occurred with movies that have more than one genre. For example, a movie could be listed as "Comedy, Action" in one place and "Action, Comedy" in another, causing the dataset to treat these as different categories. This inconsistency lead to problems when trying to understand common genres because the order and grouping shouldn't change how genres are counted.

To solve this problem and make sure each genre is consistently represented, a function was created to clean up the genre entries

The function starts by splitting the genre string wherever there's a comma. This breaks the single string into individual genre names. The function then removes any spaces around the genre names that were left after splitting. This step ensures all genre names are neat and uniform. After cleaning, the genres are counted in a way that treats the same genres the same, no matter their order in the list.

Drama and Comedy are the top two genres, indicating a strong preference for these types of films within the dataset. This suggests that these genres are not only popular among audiences but also common in production. Genres like News, Reality-TV, and Short films have very few entries. This could indicate that these genres are less commonly produced or perhaps less likely to be included in a dataset focused on mainstream cinema.

A set of genres that each have fewer than 500 entries in the dataset have been identified. These genres include:

Music,Animation,History,Sport,War,Musical,Western,Short,News,Reality-TV The decision to exclude these genres are because including categories with small numbers of samples can potentially lead to overfitting on rare categories as the model might end up memorizing these examples rather than learning to generalize from broader patterns.

## Preparing data for training

### Genre Encoding

The MultiLabelBinarizer from sklearn.preprocessing module was utilized to encode the genres into a binary format. This makes the predictor variable compatible with the neural network.

### Image Preprocessing for Model Input

Preprocessing Steps:

1. Setting Target Image Size

To ensure consistency in input size, all images are resized to a uniform dimension. In this case, each image will be resized to 112x112 pixels. This dimension is chosen to balance between retaining enough image detail and managing computational efficiency.

1. Loading and Resizing Images

Using the load\_img function from the keras.preprocessing.image module, each image is loaded from its file path and resized to the specified dimensions.

1. Converting Images to Arrays

After resizing, each image is converted into a NumPy array using the img\_to\_array function. This transformation is necessary for the images to be processed and utilized by neural network models.

1. Normalization

Pixel values of images, originally ranging from 0 to 255, are normalized to a scale of 0 to 1. This normalization is performed by dividing the array values by 255.0. Normalizing the data helps in speeding up the convergence during training and improves model performance.

1. Storing Preprocessed Images

Each preprocessed image array is appended to a list. Once all images have been processed, this list is converted into a single NumPy array. This final array, X, holds all the image data in a format that is ready for model training.

### Data Splitting and Augmentation for Model Training

To effectively train, validate, and test our models, the dataset was split into three subsets: training, validation, and testing sets. The dataset is initially split into a training set and a temporary set using the train\_test\_split method from sklearn.model\_selection. 80% of the data was allocated to the training set and 20% to the temporary set, ensuring a random shuffle of data points to avoid any bias related to the order in the dataset. The temporary set is further divided equally into validation and test sets. Each set now contains 10% of the original data. Furthermore, data augmentation was used to increase the diversity of the training set by applying random transformations (like rotation, shifting, zooming). This helps prevent overfitting and allows the model to generalize better to new, unseen data. ImageDataGenerator from keras.preprocessing.image was used for on-the-fly image augmentation. Configurations include:

* Rotation of up to 40 degrees
* Width and height shift up to 20%
* Shear intensity of 20%
* Zoom up to 20%
* Horizontal flipping of images
* fill\_mode set to 'nearest' to fill in new pixels that can appear after a transformation

The train\_generator object is created to automate the process of fetching batches of augmented images during model training. This ensures that the model sees slightly different versions of input data at each epoch, enhancing its ability to generalize.

## Designing the Models

For the project, two different models are designed to handle the classification of movie genres based on their posters: a custom-built model and a pre-trained model utilizing transfer learning. Each model's architecture, hyperparameters, and the rationale for specific choices are detailed below.

### Custom-Built Model

#### Architecture

Convolutional Layers: The model consists of multiple convolutional layers with varying numbers of filters (128, 64, 32). Each convolutional layer uses a 3x3 kernel size.

Activation Function: 'ReLU' (Rectified Linear Unit) is used for its ability to provide non-linear transformation and improve convergence during training.

Batch Normalization: Applied after each convolutional layer to stabilize and speed up the neural network training.

Pooling Layers: MaxPooling2D with a 2x2 window is used to reduce the spatial dimensions of the output volume.

Dropout Layers: Placed strategically to prevent overfitting by randomly setting a fraction of input units to 0 at each update during training time.

Dense Layers: Towards the end, flattened output is passed through dense layers with 256 and 128 units respectively before reaching the final layer.

Output Layer: A dense layer with 14 units (one for each genre) uses a 'sigmoid' activation function to handle the multi-label classification scenario.

#### Reasoning

The model is structured to gradually reduce the spatial dimensions while increasing the depth, enhancing the network’s ability to learn increasingly abstract features. Dropout and batch normalization are crucial for enhancing the model's generalizability.

### Pre-Trained Model (VGG16)

#### Configuration

Base Model: VGG16, loaded with pre-trained weights from ImageNet. The top layers are not included, allowing custom layers to be added for the specific task.

Trainable Layers: While most of the VGG16 layers are frozen to retain the learned features, the last convolutional layer (block5\_conv3) is set to trainable to allow fine-tuning on more abstract features specific to movie posters.

Additional Layers: Post VGG16 base, the model includes a Flatten layer, followed by two Dense layers with dropout for regularization. The final layer mirrors the custom model with a 'sigmoid' activation function for binary-classification problems.

#### Choice of Pre-Trained Model

VGG16 is chosen for its simplicity and effectiveness in image classification tasks. It's sufficiently deep to extract useful features but not so complex as to require extensive computational resources, making it ideal for fine-tuning.

Feature Extraction vs. Fine-Tuning

The VGG16 model is primarily used for feature extraction here, with minimal fine-tuning. This strategy leverages the pre-trained network’s ability to recognize image features and only adjusts the final layers to make predictions specific to the dataset at hand.

### Implementation Notes

Two versions of each model type are discussed:

Standard Models: These utilize the normal training dataset.

Augmented Models: These models are trained on an augmented dataset (using data augmentation techniques such as rotations, shifts, and flips) to improve robustness and reduce overfitting.

## Results

The results from training the custom CNN models and the VGG-based models, both with and without data augmentation, offer insightful data about their performance across 30 epochs. The hyperparameters for the models was the learning rate being 0.001 the batch size 32 and the number of epochs was 30. The reason why this was done is because these hyperparameters offer a balance between speed and accuracy

### Custom CNN Model

Standard CNN Performance:

At epoch 1, the loss was relatively high at 0.9090, and the binary accuracy started at 51.47%. This indicates the model initially guessed near random but slightly better, typical for the start of training.

The model stopped training at epoch 30, the loss significantly decreased to 0.3708, and the binary accuracy improved to 85.25%. The validation loss and accuracy settled at 0.4096 and 83.57%, respectively, showing good model generalization from training to validation data.

CNN with Augmentation (CNN-AUG):

Began with a loss of 0.9177 and a binary accuracy of 51.36%. The initial validation results were poorer than the non-augmented model, with a loss of 0.8191 and binary accuracy of 43.92%.

Improved markedly throughout the training, finishing at a loss of 0.3808 and binary accuracy of 84.86% at epoch 28. The final validation loss was 0.3920 with a binary accuracy of 84.41%, indicating effective learning and adaptation, slightly better than the non-augmented model.

VGG-Based Model

Standard VGG Model:

Starting Performance: Commenced with a lower initial loss of 0.4730 and a high binary accuracy of 84.03%, demonstrating the strength of transfer learning right from the outset.

Ending Performance: By epoch 19, the model showed a loss of 0.2256 and a binary accuracy of 88.28%. Validation loss slightly increased to 0.3866 with binary accuracy peaking at 85.43%. This model showed robust performance quickly, stabilizing with high accuracy.

VGG with Augmentation (VGG-AUG):

Starting Performance: Opened with a loss of 0.3790 and binary accuracy of 84.84%. The initial validation results were strong, showing a loss of 0.3648 and binary accuracy of 85.13%.

Ending Performance: By epoch 15, the loss was reduced to 0.3043 and binary accuracy increased to 87.33%. The validation loss was 0.3559 with a binary accuracy of 85.70%, displaying excellent generalization.

### Prediction

The real-world application of these models is demonstrated by their predictions on three distinct movies: "Beekeeper," "Dune," and "Insidious." Each has a known genre combination, providing a benchmark against which the models' genre classification accuracy can be evaluated.

"Beekeeper" - Actual Genres: Action, Thriller

* CNN Models: Both predicted 'drama', which is incorrect.
* VGG Model: Predicted 'drama', still incorrect.
* VGG Augmented Model: Predicted 'drama', again incorrect.

"Dune" - Actual Genres: Action, Adventure, Drama

* CNN Models: Made no genre predictions, indicating a failure to detect any of the relevant genres.
* VGG Model: No predictions, similar failure as CNN models.
* VGG Augmented Model: Predicted 'comedy', which is entirely inaccurate.

"Insidious" - Actual Genres: Horror, Mystery, Thriller

* CNN Models: Both versions predicted 'drama', which is not among the actual genres.
* VGG Models: Standard VGG made no predictions, and the augmented VGG predicted 'drama', both failing to recognize the correct genres.

## Evaluation

Evaluation metrics that were used are:

Precision is the ratio of correctly predicted positive observations to the total predicted positives. High precision relates to the low false positive rate. Recall is the ratio of correctly predicted positive observations to all observations in the actual class. High recall relates to the low false negative rate. F1-Score is the weighted average of Precision and Recall. This score takes both false positives and false negatives into account. AUC-PR is a valuable metric in imbalanced datasets as it combines precision and recall into a single measure, reflecting the ability of a classifier to handle positive (minority) classes effectively Confusion matrices provide a breakdown of true positives, false positives, true negatives, and false negatives, critical for understanding model performance in terms of type I and type II errors.

### Custom CNN Models

Struggle with precision and recall for most genres, with many genres with few samples such as biography, documentary, and fantasy have zero scores in almost all metrics across models, indicating difficulties in the model's ability to learn features specific to these categories. The CNN models performs moderately with comedy and drama achieving the highest AUC-PR scores, indicating better handling of these genres compared to others. However, genres like family, documentary, and sci-fi show particularly low scores, suggesting significant room for improvement in recognizing these categories.

VGG-Based Models

Generally, provide better precision and recall scores. Particularly, the VGG augmented model shows improved performance in both metrics, especially in drama and comedy, indicating effective learning and generalization capabilities enhanced by data augmentation. Data augmentation has shown to improve the performance of the VGG model significantly, as seen in the higher binary accuracy and better f1-scores in specific genres. For the CNN model, however, augmentation didn't make a significant difference. The VGG models, especially the augmented one, generally provide better outcomes in terms of loss, accuracy, and across most genres' precision and recall. The higher capability of VGG in feature extraction and adaptability to the augmented data showcases its potential in handling complex image classification tasks like this one. The VGG augmented model tops the AUC-PR scores among all, particularly excelling in comedy and drama but also showing remarkable scores in genres typically challenging for the CNN models, such as action and romance. The increases in scores for genres like documentary and fantasy underscore the benefits of augmentation in helping the model learn more complex or less frequent patterns.

All models consistently identify non-existence of genres well (high true negatives), possibly due to the skewed nature of the data towards negative cases. True positives are notably low across most genres and models, with some exceptions in comedy and drama for the VGG augmented model, reflecting its ability to detect more nuanced features of these genres. Almost all models completely fail to detect any true positives in several underrepresented genres like documentary, biography, and sci-fi, which could be a consequence of insufficient training examples.

## Discussion

Both augmented models showed slightly better or comparable validation accuracy by the end of training compared to their non-augmented counterparts, highlighting the benefits of using augmented data for enhancing model generalizability.

The VGG-based models consistently outperformed the custom CNN models in terms of accuracy, demonstrating the power of transfer learning for this application. They achieved higher accuracy and stabilized faster than the custom-built models.

The VGG models, particularly the augmented version, showed more stable and consistent improvement in validation accuracy, suggesting a better handling of overfitting.

The poor performance on less frequent genres across all models suggests a need for strategies that could improve learning in these areas, possibly through techniques like SMOTE for synthetic data generation or potentially gathering more data.

It may be beneficial to experiment with other pre-trained models or tweak the network architectures, training processes, and augmentation techniques to find optimal settings for all genres, not just the predominant ones.