**Movie Genre Classification System Report**

**Zain Khan (29217) | Sohaib Sohail (28994) | Barkhi Khan (29261)**

**Project Overview**

Our project tackles the challenge of automatically identifying movie genres from poster images using deep learning. We built a Convolutional Neural Network that can look at a movie poster and predict what genres it belongs to - whether it's an action film, a comedy, or maybe both. This multi-label approach makes sense because most movies don't fit into just one category (think "Guardians of the Galaxy" being both sci-fi and comedy).

**How We Built the System**

**Our Data Processing Workflow**

We organized our work into five main Python scripts, each handling a specific part of the process:

1. **Getting the Data (getdata.py)**: We wrote code to automatically download thousands of movie poster images from web URLs. Since downloads can fail, we added retry logic and error handling to make sure we got as many images as possible.
2. **Organizing everything (organisedata.py)**: The original movie data had genres listed like "Action, Comedy, Drama" which isn't great for machine learning. We converted these into separate binary columns - so each movie gets a 1 or 0 for each possible genre.
3. **Cleaning Up (datacleaner.py)**: Not every movie poster download worked, so we had to remove movies from our dataset where the image was missing. This kept everything consistent.
4. **Preparing Labels (preparelabels.py)**: Before training, we double-checked that our data was properly formatted and split into training, validation, and testing sets.
5. **Training the Model (cnnmodel.py)**: This is where the actual CNN gets built, trained, and evaluated.

**Our CNN Design**

We decided to build our own CNN from scratch rather than using a pre-trained model. Here's what we came up with:

* **Input**: Takes movie posters sized at 182×268 pixels (we chose these dimensions to match typical poster proportions)
* **Feature Extraction**: Three convolutional layers that get progressively deeper (32, 64, then 128 filters), each followed by max pooling
* **Classification**: A fully connected layer with 128 neurons, then the final output layer with 28 neurons (one for each genre)

**Why Custom Dimensions?** Most pre-trained models need 224×224 pixel images as a minimum, but we found that 182×268 works better for movie posters since it preserves their natural rectangular shape. Plus, smaller images mean faster training.

**Preventing Overfitting**

We used several tricks to make sure our model generalizes well:

* **Data augmentation**: Randomly flip, rotate, and zoom images during training
* **Dropout**: Randomly turn off 50% of neurons during training
* **Early stopping**: Stop training if the validation performance stops improving

**Technical Stack**

We relied on several key libraries:

* **TensorFlow/Keras**: Our main deep learning framework
* **NumPy & Pandas**: For data manipulation and number crunching
* **Scikit-learn**: For splitting data and calculating performance metrics
* **Requests**: For downloading images from the web

**Implementation Details**

**Multi-Label Classification**

Unlike typical image classification where you pick one category, we needed to handle multiple genres per movie. We used sigmoid activation (instead of softmax) so each genre gets its own independent probability score. Binary crossentropy loss lets each genre be predicted separately.

**Image Processing**

Every poster goes through the same preprocessing steps:

1. Load and decode the JPEG file
2. Resize to our standard 182×268 dimensions
3. Normalize pixel values to the 0-1 range
4. Apply random augmentations during training

**Training Setup**

We used pretty standard settings:

* Adam optimizer with 0.001 learning rate
* Batch size of 32 images at a time
* 60% of data for training, 20% for validation, 20% for final testing
* Track accuracy, precision, and recall metrics

**Results and Evaluation**

Our model provides detailed performance metrics including overall accuracy and per-genre breakdowns. We use a 0.5 threshold to convert probability scores into yes/no predictions for each genre.

**Why This Approach Works**

Our custom CNN strikes a good balance between complexity and performance:

* **Tailored dimensions**: 182×268 pixels fit movie posters better than square formats
* **Progressive learning**: Each convolutional layer learns increasingly complex visual patterns
* **Efficient processing**: Smaller input size means faster training and lower memory usage
* **Real-world ready**: Handles 28 different genres simultaneously

**Deployment**

The final trained model gets saved as "movegenreclassifier.h5" along with helper functions that can:

* Process new poster images
* Generate genre predictions with confidence scores
* Display results in a readable format

This whole pipeline demonstrates how you can go from raw data (CSV files and image URLs) all the way to a working movie genre classifier that could actually be useful in real applications like movie recommendation systems or content organization tools.

This report was generated with the help of AI for better explanation of our code