Movie Genre Prediction Using Multi-Label Image Classification

Presented By Group - 12

The Team

Adithya Nair

AM.EN.U4CSE19103

Akshay Hari

AM.EN.U4CSE19104

Sreejith K Pai

AM.EN.U4CSE19153

Table of Contents

01

Background

Brief Introduction & Aim of our Project

Posters are more than just a promotional material, that captures a viewer’s attention.

It is a reflection of how the audience is able to perceive an image about the movie.

Therefore a model which can extract the features of a movie poster and identify the genres can become handy for both the film-makers as well as the designers.

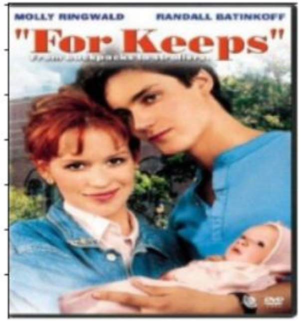
02

Problem Definition

A Model to Predict Movie Genres

Problem definition

Movie posters cannot be labelled to just one pinpoint genre. Thus, determining the genre of a Movie poster is a Multi-label classification problem. The objective is to solve this problem using Neural Networks.



→ Drama / Romance

03

Related Works

Related Works

Identifying Multiple Objects in an Image

Early work from Barnard and Forsyth focused on identifying objects in particular sub-sections of an image.

Multi-Label Classification of Satellite Images

Work from Daniel Gardner & David Nichols focused on classification of Satellite Images using multiple models.

04. Contributions

- A model for multi label image classification using Feed Forward neural network architecture.
- Used different loss functions including Cross Entropy, BCELoss etc.
- Tried different pre-trained CNN models like Resnet-50, VGG-16, InceptionV3, EfficientNet etc.
- Used different optimizers like Adam, SGD etc.

05

Solution Approach

Neural Network Models

Neural Network models

Feed Forward Neural Network

Convolutional Neural Network

Data Preprocessing

Data Preparation

Image layer

Images with 3 RGB channels with varying sizes

Dimension

Resize all images to 350*350 dimension for easy training and efficiency

Splitting

Splitting dataset into train (85%) and test (15%)

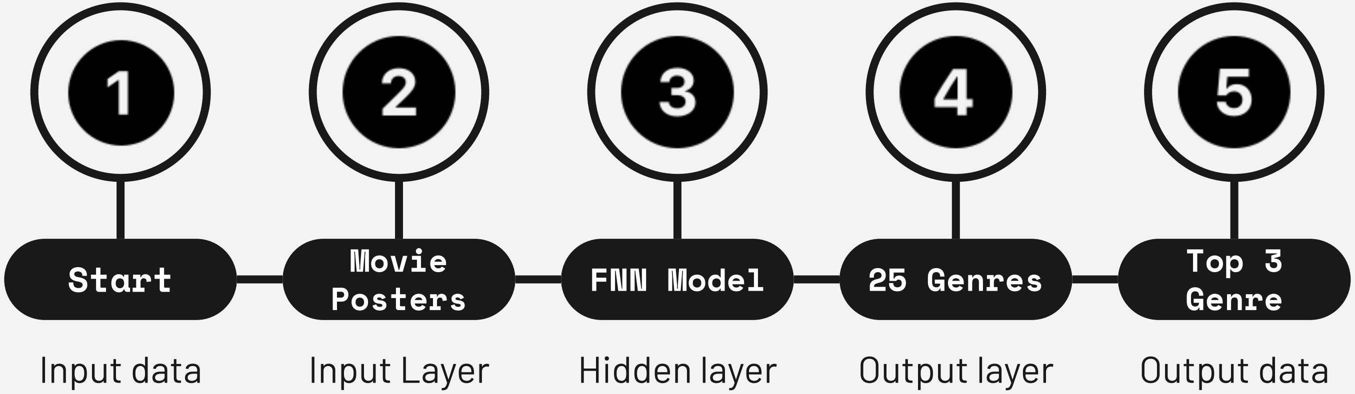
Dataloader

Load images into training and test data loader with batch size of 33.

Feed Forward Neural Network

Model

- Each layer consist of neurons which are triggered based on activation functions.
- Architecture consist of single linear layer , three hidden layer and an Output layer.
- **ReLU activation function** is used for input and hidden layers
- Output layer is activated using **Sigmoid function**
- Sigmoid function converts each value into probability score between 0 and 1



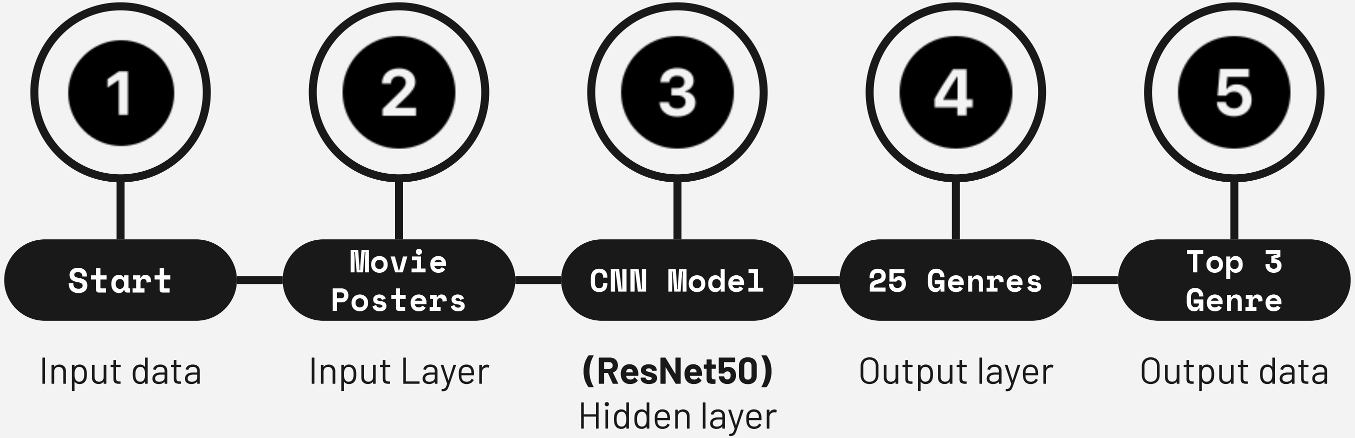
Implementation of FFN

```
class FirstModel(Base):

    def __init__(self):

        super().__init__()
         # input layer
        self.linear1 = nn.Linear(input_size, hidden_size[0])
         # hidden layer
        self.linear2 = nn.Linear(hidden_size[0], hidden_size[1])
        self.linear3 = nn.Linear(hidden_size[1], hidden_size[2])

        self.linear4 = nn.Linear(hidden_size[2], output_size)

    def forward(self, xb):

        # Flatten images into vectors
        out = xb.view(xb.size(0), -1).requires_grad_()
        # Get intermediate outputs using hidden layer
        out = self.linear1(out)
        # Apply layers & activation function
        out = F.relu(out)
        out = self.linear2(out)
        out = F.relu(out)
        out = self.linear3(out)
        out = F.relu(out)

        # Get predictions using output layer
        out = self.linear4(out)

        y_pred = torch.sigmoid(out)

        return y_pred
```

Convolutional Neural Network - ResNet50

Model

- Based on pre-trained CNN Resnet50 Architecture..
- Solves problem of Vanishing/Exploding of gradients.
- Introduction of skip connection for freezing the intermediate layers.
- Capable of learning deeper networks in an optimised way by reducing the no. of parameters.
- Sigmoid function converts each value into probability score between 0 and 1



Implementation of CNN

```
resnet = models.resnet50(pretrained=True)
```

```
for param in resnet.parameters():
    param.requires_grad = False
```

```
in_features = resnet.fc.in_features
resnet.fc = nn.Linear(in_features, num_classes)
```

```
resnet = resnet.to(device)
opt = optim.Adam(resnet.parameters(), lr=0.0001)
loss_fn = nn.BCELoss()
```

```
outputs = resnet(inputs)
outputs = torch.sigmoid(outputs)
```

Loss Function

Model

- We used Binary Cross Entropy Loss (BCE Loss).
- Used along with Sigmoid function
- Compares each of the predicted probabilities to actual class output which can be either 0 or 1.
- Calculates the score that penalizes the probabilities based on the distance from expected value.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Binary Cross-Entropy / Log Loss

06

Dataset

Movie Poster Dataset

7867

Number of Image Data Samples

Dataset

IMDB Data

1980-2015

Our dataset is taken from IMDB collection of movie posters.

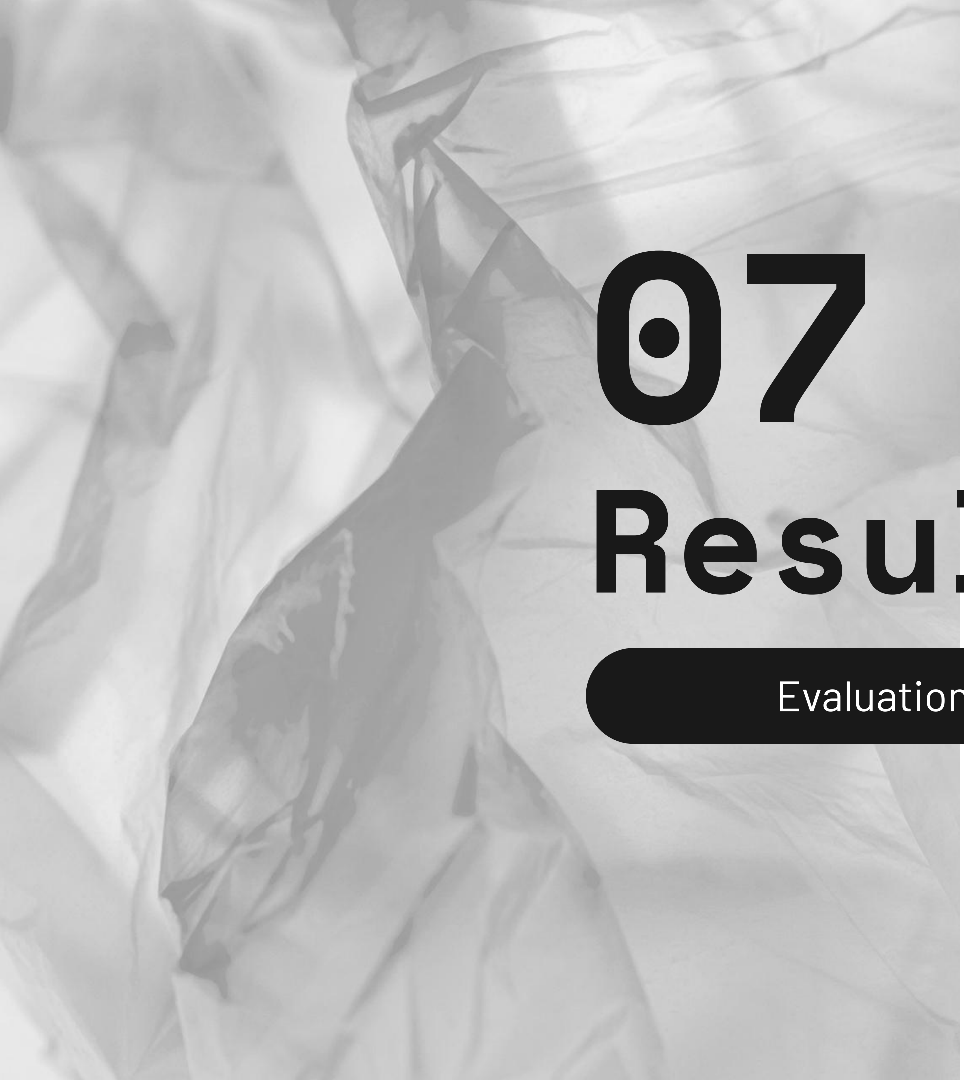
Features

27 Columns

Features include ID of images & labels along with their corresponding genres.

Movie Poster Dataset

	Id	Genre	Action	Adventure	Animation	Biography	Comedy	Crime	Documentary	Drama	Family	Fantasy	History	Horror
0	tt0086425	['Comedy', 'Drama']	0	0	0	0	1	0	0	1	0	0	0	0
1	tt0085549	['Drama', 'Romance', 'Music']	0	0	0	0	0	0	0	1	0	0	0	0
2	tt0086465	['Comedy']	0	0	0	0	1	0	0	0	0	0	0	0
3	tt0086567	['Sci-Fi', 'Thriller']	0	0	0	0	0	0	0	0	0	0	0	0
4	tt0086034	['Action', 'Adventure', 'Thriller']	1	1	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7249	tt2409818	['Action', 'Crime', 'Thriller']	1	0	0	0	0	1	0	0	0	0	0	0
7250	tt2062622	['Animation', 'Comedy', 'Family']	0	0	1	0	1	0	0	0	1	0	0	0
7251	tt2442502	['Comedy']	0	0	0	0	1	0	0	0	0	0	0	0
7252	tt3455850	['Documentary']	0	0	0	0	0	0	1	0	0	0	0	0
7253	tt4179482	['Animation']	0	0	1	0	0	0	0	0	0	0	0	0



07

Result

Evaluation & Observations

Evaluation

- The training as well as the testing datasets were evaluated separately.
- The probability scores obtained as output after **Sigmoid activation** in the final layer, were converted into one-hot encoded values based on a condition.
- The top three probability scores were converted into 1's and the remaining scores were converted into 0's.
- This ensured prediction of three top genres for each sample.
- The accuracy rates were then calculated based on the predicted outputs and the actual labels (1's at the same index of both the outputs were only included).

Accuracy Rates

21.93%

Feed Forward with Multiple Layers

60.21%

CNN using ResNet50 model

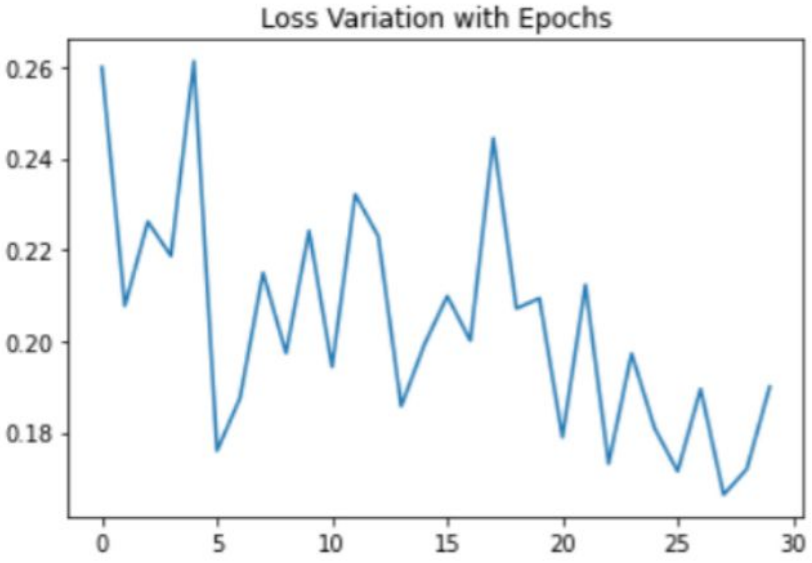
Feed Forward Neural Network - Observations

```
TRAIN LOSS:  0.2542370557785034
Epoch [0], val_loss: 0.2555, val_acc: 21.9061
TRAIN LOSS:  0.2609313726425171
Epoch [1], val_loss: 0.2506, val_acc: 21.9506
TRAIN LOSS:  0.2470485270023346
Epoch [2], val_loss: 0.2505, val_acc: 21.9108
TRAIN LOSS:  0.23429888486862183
Epoch [3], val_loss: 0.2481, val_acc: 21.9342
TRAIN LOSS:  0.24423915147781372
Epoch [4], val_loss: 0.2467, val_acc: 21.9310
```

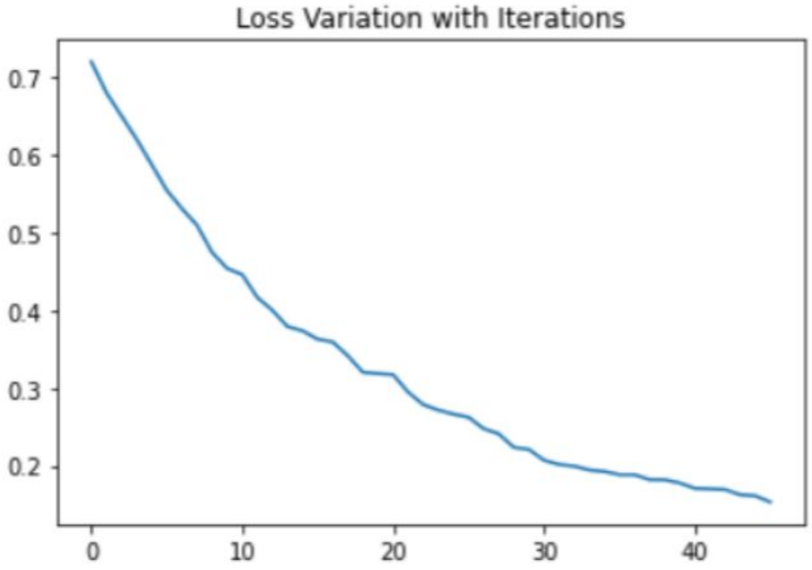
Convolutional Neural Network - Observations

```
Epoch: 24/30, Test acc: 59.93, Train acc: 61.28
CORRECT: 1507
TOTAL: 2513
CORRECT: 8821
TOTAL: 14361
Epoch: 25/30, Test acc: 59.97, Train acc: 61.42
CORRECT: 1513
TOTAL: 2513
CORRECT: 8841
TOTAL: 14361
Epoch: 26/30, Test acc: 60.21, Train acc: 61.56
CORRECT: 1493
TOTAL: 2513
CORRECT: 8836
TOTAL: 14361
Epoch: 27/30, Test acc: 59.41, Train acc: 61.53
CORRECT: 1495
TOTAL: 2513
CORRECT: 8848
TOTAL: 14361
Epoch: 28/30, Test acc: 59.49, Train acc: 61.61
CORRECT: 1493
TOTAL: 2513
CORRECT: 8900
TOTAL: 14361
Epoch: 29/30, Test acc: 59.41, Train acc: 61.97
```

Convolutional Neural Network - Observations



Graph comparing the change in Loss function with respect to the number of Epochs.



Variation of Loss Function with increase in number of iterations.

08. Hyper parameter Tuning

Strategies

- The gradient descent optimizers used to update the model parameter are :
- Stochastic gradient descent with a learning rate of 0.05 in FNN.
- Adam Gradient Optimization with a learning rate of 0.0001 in CNN.

Activation Function

- ReLu activation function was used in hidden layers of FNN model
- Sigmoid function used in the output layer of both FNN and CNN model

09. Comparisons

- Below 4 CNN Architectures were applied :

Model	Test Accuracy	Training Accuracy
VGG-16	54.98%	60.78%
ResNet50	60.21%	61.56%
InceptionV3	46.27%	46.85%
EfficientNet	45.19%	74.48%

- ResNet was the best model with an accuracy of 60.21 on test dataset and 61.56 on train dataset.

References

- Multi-label Classification of Satellite Images with Deep Learning by Daniel Gardner, David Nicols

- Movie Posters' Classification into Multiple Genres.
  Vaibhav Narawade,Aneesh Potnis,Vishwaroop Ray

- Residual Attention: A Simple but Effective Method for Multi-Label Recognition.·Ke Zhu, Jianxin. Wu.

- Research on Multi-label Clothing Image Classification
  Based on Convolutional Neural Network.
  Ying Hong, DOI:10.1007/978-3-030-63784-2-1,05

- Movie genre classification using convolutional neural networks. Luka Popovic,Santiago Cepeda,Nino Scherrer

Thank You