Anime Genre Classification Using CNNs

Megan Chen myc9078@nyu.edu

Mimi Chen xc2063@nyu.edu

Jack Lee jl11862@nyu.edu

Milestone 2

1. Methodology

1. **Data Source and Augmentations** We will use a Kaggle Dataset from 2023 consisting of 24,905 posters. Each poster has an id, name (in original language), English name, other name, rating of anime, genres of the anime, synopsis, type (movie or tv series), number of episodes, and date aired.

   (a) *Image Processing*
   The crucial first step in our methodology is the preprocessing of these images to ensure uniformity and compatibility with our models. This preprocessing pipeline encompasses various stages, from image resizing and normalization to consistency checks and categorical data transformation. We build a function to process each image obtained via a URL, which perform two essential tasks:

      i. Resizing: Each image is resized to a uniform dimension of 224x224 pixels.

      ii. Normalization: Post-resizing, we normalized the pixel values for each image. The pixel values are scaled to a range from 0 to 1 (from the standard range of 0 to 255). The normalization aids in numerical stability for our learning operations.

   (b) *Error Handling*
   In the case that an image is inaccessible (non-200 HTTP status) or an exception occurs, we return a placeholder image that is a zero-valued array (black image) of the same target size. This is designed to ensure that the data set remains consistent for batch processing in our training stage.

   (c) *Consistency Check and Filtering*
   We performed a consistency check to ensure that all processed imaged that do not conform to the standard size are calculated and reported. Following this check, we filter the data set to exclude the images that do not match the desired dimension of 224x224x3.

   (d) *Genre One-Hot Encoding*
   The final step of our data processing is transforming the categorical genre data associated with each anime image. The genres are converted into a binary matrix representation through one-hot encoding. This converts categorical labels into a format that is suitable for computation. For each unique category in the dataset, we generate a new binary column representing the presence (1) or absence (0) of the category.

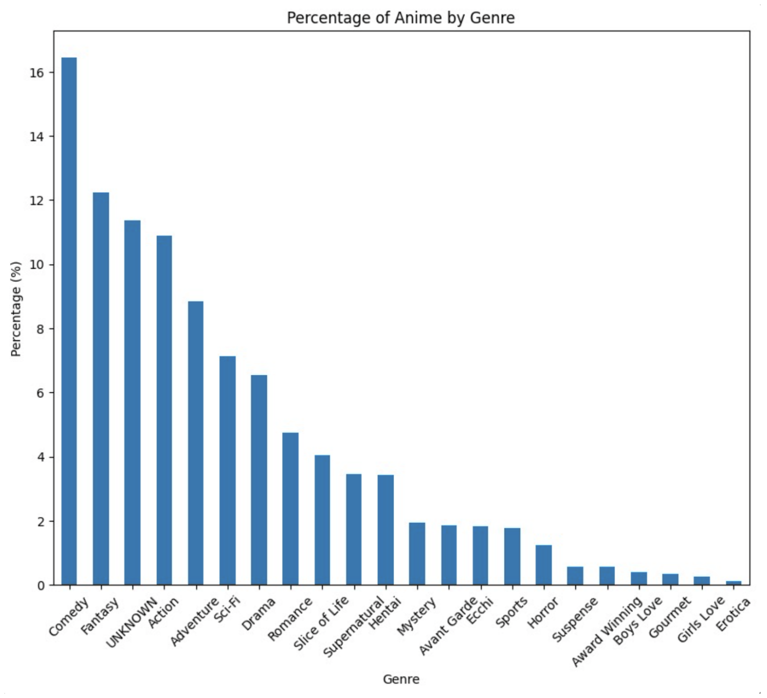

Figure 1: Class Distribution of Anime Dataset

2. **Proposed Model**

   We have used two different deep learning models (shown below) to classify the genre for multi-label classification based on the anime poster. The process of feature extraction was done using a convolutional neural network (CNN). CNNs are a class of deep learning algorithms that are primarily used for processing grid-like data such as images, which makes them adept at automatically detecting and learning hierarchical patterns in data. The core components of a CNN include convolutional layers, pooling layers, and fully connected layers, all critical to the process of feature extraction and classification.

3. **Proposed Model Architecture**

   The implemented architecture of our model is constructed using the Sequential model from tensorflow.keras.models, consisting of convolutional layers, flattening layers, and a fully connected layer.

   (a) Convolutional Layers:

   The first layer is a convolutional layer with 32 filters of size 3x3, followed by a 2x2 max-pooling layer. The second layer has 64 filters, and the third layer has 128 filters, each followed by the same max-pooling layer configuration.

   (b) Flattening Layer:

   Post convolutional layers, a flattening layer is employed to transform the 2-dimensional matrix into dimensional vector

(c) Dense Layer:

A fully connected layer with 128 neurons and the ReLU activation function is integrated to transform features extracted by previous layers. The output layer consists of neurons equal to the number of classes, using a sigmoid activation function to suit the nature of multi-class classification.

4. **Learning Phase**

Referencing the process of Hossain et.al., our system consists of two states: training and testing.

(a) *Training State*

i. The initial step of our Convolution Neural Network is converting the anime posters into a feature space to perform feature extraction. Our proposed model consists of seven layers. The network begins with a convolutional layer equipped with 32 filters of 3x3 size and a ReLU activation function, immediately followed by a 2x2 max pooling layer, which helps in reducing the spatial size of the output and controlling overfitting. This pattern is repeated twice more: the second convolutional layer has 64 filters, and the third one has 128 filters, each followed by a respective 2x2 max pooling layer. These layers are instrumental in feature extraction and capturing spatial hierarchies. After the convolutional and pooling layers, a flattening layer is introduced to transform the 2D feature maps into a 1D feature vector. This vector is then fed into a dense layer with 128 neurons, again using a ReLU activation function, serving the purpose of learning non-linear combinations of features. The architecture culminates in an output layer, which is a dense layer with a number of neurons equal to the number of classes in the dataset, using a sigmoid activation function for multi-label classification. This structured layering in the CNN allows for effective feature extraction and classification in image-based datasets.

ii. For extracting features, the operation in the convolution layer is

$$F_{k,x,y} = \text{ReLU}\left(\sum_{i=-1}^{1}\sum_{j=-1}^{1}\sum_{c=1}^{3} I_{x+i,y+j,c} \cdot K_{k,i,j,c} + b_k\right) \tag{1}$$

iii. The operation performed by the max pooling layer is represented as

$$F_{x,y} = \max\left(I_{2x:2x+1,2y:2y+1}\right) \tag{2}$$

iv. The first dense layer is a fully connected layer with 128 neurons and ReLU activation

$$h_i = \text{ReLU}\left(\sum_{j} w_{ij} \cdot x_j + b_i\right) \quad \text{for } i = 1, 2, ..., 128 \tag{3}$$

v. Finally, the output layer is also a fully connected layer, with the number of neurons equal to the number of classes, and uses the sigmoid activation function. The operation for each output neuron is:

$$y_k = \sigma\left(\sum_i w_{ki} \cdot h_i + b_k\right) \quad \text{for } k = 1, 2, ..., N \tag{4}$$

In the compilation phase of the neural network, we configure the model with the necessary specifications for the training process. The model is compiled using the Adam optimizer, which is an extension to stochastic gradient descent. Throughout the training epochs, the Adam optimizer adjusted the weights to minimize the binary cross-entropy loss. Simultaneously, the Hamming loss was monitored as a performance metric, providing insight into the proportion of labels that are misclassified.

(b) *Testing State*

We use the Hamming loss to measure how many samples were incorrectly classified. It calculates the number of label mismatches between the true and predicted labels for each instance, averages it over all instances, and then normalizes by the number of labels.

$$\text{Hamming Loss} = \frac{1}{N}\sum_{i=1}^{N} \frac{\text{XOR}(y_{\text{true}_i}, y_{\text{pred}_i})}{L} \tag{5}$$

5. **Model Evaluation**

We evaluate the performance of our custom model, as well as the performance of a ResNet50 pre trained on ImageNet, with 10 epochs (limited due to computational resources), and a learning rate of 0.001.

2. Results

Custom Model

The Hamming Loss on the test set is 0.094, meaning that on 9.4% of the predicted labels across all the predicted labels across all the test set instances were incorrect. This level of accuracy demonstrates the model's capability in handling the multi-label classification task, yet it also highlights the opportunity for further refinement to reduce the rate of misclassification.

ResNet 50

In comparison, the well-established ResNet 50 model was employed as a benchmark. When evaluated on a subset of 1000 images from the test set, the ResNet 50 model yielded a Hamming Loss of 0.133. For the full dataset, the Hamming Loss was 0.075. This lower score on the full dataset indicates a better performance, with fewer misclassifications when trained on more data.

3. Analysis

Our custom model displayed a commendable performance on the multi-label classification task, as quantified by the Hamming Loss. The analysis of the Hamming Loss across both

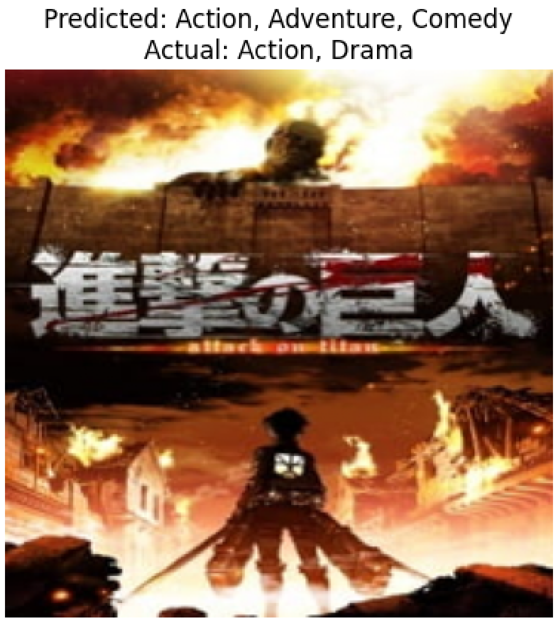

Figure 2: Sample Outcome

the custom model and the ResNet 50 provides insightful perspectives on their performance in multi-label classification tasks. A Hamming Loss of 0.094 for the custom model and 0.075 for our pretrained ResNet50 model. Since both of these results were obtained from models trained and tested on the 1000 instance subset, we have yet to determine how our model will perform on the full dataset. The custom model, despite its simpler architecture compared to ResNet 50, showed better performance on our established metric. This competence could be attributed to specific attributes of the model that can be stronger in the circumstances where ResNet50 does not perform to its full potential. Since the ResNet50 is a dimensionally deep network with 50 layers, the rather simple nature of anime art shows that the simpler model might generalize better on the test data. Capturing a wider range of features, some of which may not be relevant to anime art may lead to the model learning irrelevant patterns that decrease performance.

4. Additional Analysis

For milestone 2, we wanted to test different parameters to build our custom model. Therefore, for both the ResNet50 pre-trained on ImageNet and our custom model, we used a subset of 1000 images instead of the entire dataset. In milestone 3, we plan to deploy this model to the entire dataset to achieve a more comprehensive result. Additionally, we plan to evaluate the performance of our model through looking at other performance metrics such as recall, precision, and F1 score in additon to Hamming Loss for a fuller understanding of our results. We also plan on adjusting the learning rates to see if the loss values change at different rates and determine the best parameters for our model. Another implementation we are considering is evaluating the performance of other models like VGG, Lenet, and Alexnet.

5. Work Plan

Up until now, all team members have collectively planned, discussed, and executed the initial design and implementation of our model, as well as working to determine the most optimal hyperparameters for a custom model.

Week	Megan
11/30	Evaluate the models on a validation set
12/4	Research CNN architecture and optimization
12/11	Collaborated to detemine hypermarameter optimization for simpler images
12/18	Research more benchmark CNNs to evaluate custom model

Week	Mimi
11/30	Research CNN architecture and optimization
12/4	Work on tuning custom model and hyperparameter optimization
12/11	Refine custom model, and train the model on a full dataset
12/18	Compare and evaluate performance of all models, tune custom model

Week	Jack
11/30	Research application of transfer learning to create custom model for classification
12/4	Design custom models that incorporate semi supervised learning techniques
12/11	Implement semi supervised learning and evaluate performance of model on unlabeled data
12/18	Compare and evaluate performance of all models, tune custom model

References

[1] Hussain, M., Bird, J.J., and Faria, D.R. (2019). A Study on CNN Transfer Learning for Image Classification. In *Advances in Computational Intelligence Systems* (Vol. 840, Advances in Intelligent Systems and Computing, pp. 16). Cham: Springer.

[2] Barney, G., and Kaya, K. (2019). Predicting Genre from Movie Posters. Note: Unpublished article.

[3] Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K., and Ghayvat, H. (2021). CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope. *Electronics*, 10(20), 2470.

[4] Hossain, N., Ahamad, M. M., Aktar, S., and Moni, M. A. (2021). Movie Genre Classification with Deep Neural Network using Poster Images. In *2021 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD)* (pp. 195-199). Dhaka, Bangladesh: IEEE.

[5] Unal, F. Z., Guzel, M. S., Bostanci, E., Acici, K., and Asuroglu, T. (2023). Multilabel Genre Prediction Using Deep-Learning Frameworks. *Applied Sciences*, 13(15), 8665.

[6] Wehrmann, J., and Barros, R. C. (2017). Movie genre classification: A multi-label approach based on convolutions through time. *Applied Soft Computing*, 61, 973-982.

[7] Chu, Wei-Ta, and Guo, Hung-Jui. (2017). Movie Genre Classification based on Poster Images with Deep Neural Networks. In *Proceedings of the Workshop on Multimodal Understanding of Social, Affective and Subjective Attributes (MUSA2 '17)* (pp. 39–45). New York, NY, USA: ACM.