Animated Series Genre Classification Using CNNs

Jack Lee JL11862@NYU.EDU
Mimi Chen XC2063@NYU.EDU
Megan Chen MYC9078@NYU.EDU

1. Methodology

1. **Data Source and Augmentations** We used 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.

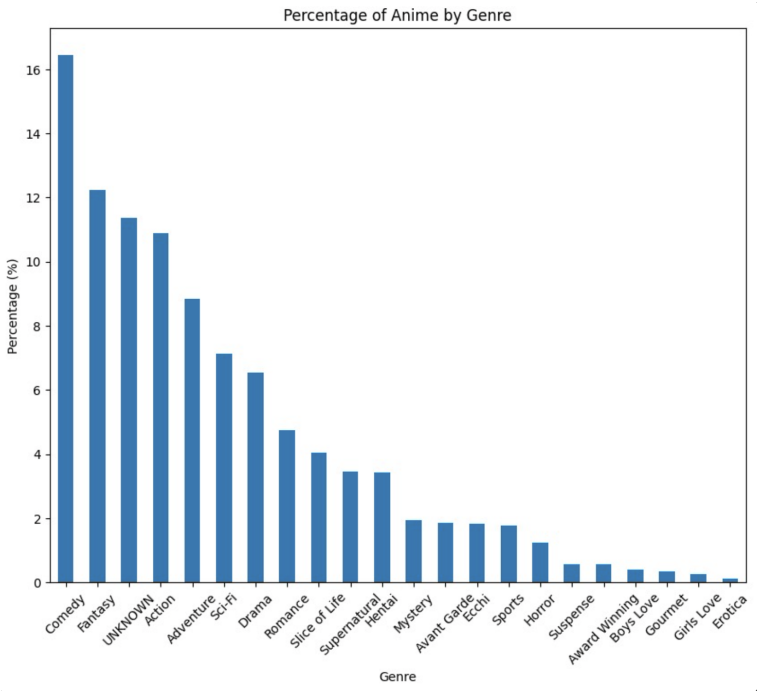

Figure 1: Class Distribution of Anime Dataset

2. Proposed Model

We have used five different deep learning models to classify the genre for multi-label classification based on the anime poster. We designed a custom model and evaluated its performance against various approaches to image classification. 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. The architecture for our base model is outlined below. Additionally, we employ semi-supervised learning on our custom CNN model to evaluate its effectiveness in classifying genres for Anime posters, alongside Transfer Learning from a pre-trained ResNet50 model. Finally, we explore new techniques in Machine Learning by utilizing Auto Keras, an open-source software library for automated machine learning that is designed to simplify the process of building and optimizing machine learning models.

3. Proposed Model Architecture

The implemented architecture of our base 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 used Hamming loss as our main metric 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. We chose this metric because of its symmetric evaluation and sensitivity to imbalanced data.

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

5. **Model Evaluation**

(a) **Base Custom CNN**

To deepen our analysis, we evaluate the performance of our base CNN model against other approaches to evaluate the most effective approach.

(b) **Custom CNN with Semi-Supervised Learning**

Due to the fact that we have unlabeled data in our dataset, we employed a semi-supervised approach for genre classification. We split our data into labeled and unlabeled subsets and the same base CNN model is trained with the labeled dataset. We then employ pseudo-labeling on the unlabeled data, where the trained model predicts genres for these posters. A confidence threshold of 0.7 is applied to identify high-confidence predictions, and these high-confidence predictions have their labels converted to binary labels, which are then combined with the initial training set. The model was then refined with newly confident predictions, progressively learning and improving its accuracy in classifying the genres of anime posters.

(c) **Transfer Learning with Pre-trained ResNet 50 Model**

Additionally, we employ a transfer learning approach utilizing the ResNet50 model. The model, pre-trained on the ImageNet dataset, serves as the foundation for this system. To tailor the model to our specific task, we modify the architecture by replacing the final fully connected layer with a new layer. This layer's output dimensions correspond to the number of genres in order to adapt the model to multi-label classification.

(d) **Transfer Learning with Semi-Supervised Learning**

In the combined methodology for our model, we integrate the principles of transfer learning with semi-supervised learning to improve the genre classification of anime posters. Our approach begins with the implementation of transfer learning using the ResNet50 architecture, then employs the same semi-supervised process used for our base model. Following the initial training, we apply pseudo-labeling to the unlabeled data. In this step, the trained model infers genre classifications for the unlabeled posters and impose the same confidence threshold of 0.7.

(e) **Exploring AutoML with AutoKeras**

Currently, Automated Machine Learning (AutoML) is continuously becoming integrated into the model development stage of Machine Learning. However, there are many challenges and limitations to AutoML. One system, AutoKeras, is particularly adept at handling the challenges associated with the design and optimization of neural network architectures, like those encountered in network morphism-based Neural Architecture Search (NAS). The process, highlighted by Jin, et.al, significantly reduces the time and expertise required to develop high-performing models. AutoKeras utilizes Bayesian optimization to guide through the search space by selecting the most promising operations each time. We employ AutoKeras to automate the finetuning process and configured to AutoKeras Image Classifier to experiment with 6 different model architectures and hyperparameter sets, ensuring some exploration depth while offsetting computational efficiency.

2. Results

1. **Base Custom CNN**

   The Hamming Loss on the test set is 0.098, meaning that on 9.8% 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.

2. **Custom CNN with Semi-Supervised Learning**

   The Custom CNN model with Semi-Supervised Learning resulted in a Hamming Loss of 0.11 on the test set, which is a marginal decrease from the base model.

3. **Transfer Learning with Pre-trained ResNet 50 Model**

The transfer learning with a ResNet 50 model yielded a Hamming Loss of 0.133. This higher score indicates worse performance compared to our first two models.

4. **Transfer Learning with Semi-Supervised Learning**

   Our model that combined the methods of transfer learning and semi-supervised learning resulted in a Hamming Loss of 0.149. This indicates that the Semi-Supervised model performed worse than the initial Transfer Learning model.

5. **AutoML with AutoKeras**

   We employed AutoKeras, an automated machine learning library, to develop a multi-label classification model. The model was then tasked with predicting a set of labels for each instance in a given test dataset. The model was trained over a series of epochs, resulting in a final training loss of 3.8906 and an accuracy of 56.65%. Upon evaluation against the test dataset, the model exhibited a loss of 5.1449 and an accuracy of 19.19%. We also achieved a Hamming Loss of 0.073.

3. Analysis

1. **Base Custom Model**

   Our custom model displayed a commendable performance on the multi-label classification task, as quantified by the Hamming Loss. This competence could be attributed to specific attributes of the model that can be stronger in the circumstances where more complex models cannot generalize as well on the data.

2. **Custom CNN with Semi-Supervised Learning**

   Our custom CNN with semi-supervised learning performed marginally worse than our base model, which could mean that the semi-supervised model is learning patterns that were not initially relevant to the classification task at hand.

3. **Transfer Learning with Pre-trained ResNet 50 Model**

   The analysis of the Hamming Loss across both the custom model and the transfer learning model provides insightful perspectives on their performance in multi-label classification tasks. The custom model, despite its simpler architecture compared to ResNet 50, showed better performance on our established metric. This difference could be a result of specific attributes of the base model that can more effectively find patterns in the circumstances. In other words, since 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. **Transfer Learning and Semi-Supervised Learning**

   Similar to the semi-supervised learning of our base model, our transfer learning model also did worse after applying semi-supervised learning. We infer that this is due to the fact irrelevant patterns are being learned.

5. **AutoML with AutoKeras**

   The model was trained over a series of epochs, resulting in a final training loss of 3.8906 and an accuracy of 56.65%. The loss value suggests that the model was moderately successful in minimizing the error over the training dataset. While the accuracy seems low at first glance, an image is only considered correctly labeled if the model correctly predicts every label for each image. Since many images in the dataset have multiple labels, the model correctly predicting every label for 56.65% of the training set is quite impressive. Of course, it also points towards a potential for considerable improvement as the discrepancy between loss and accuracy suggests that while the model has learned to correctly predict more than half of the training instances, the confidence in its predictions, as measured by the loss function, is not particularly high. Upon evaluation against the test dataset, the model exhibited a loss of 5.1449 and an accuracy of 19.19%. The pronounced increase in loss and the significant drop in accuracy compared to the training data are indicative of overfitting. However, our hamming Loss of 0.073 is the lowest of our 5 models.

4. Conclusion

Since Hamming loss represents the fraction of the wrong labels over the total number of labels, this indicates that our models are performing quite well on not incorrectly labeling posters. Although the accuracy may be low, this is because accuracy is based on correctly labeling every genre for each image. Thus, since there are so many different labels (genres) in the dataset, it is difficult for the models to correctly predict every label on the image. Additionally, we found that transfer learning on a ResNet50 model performed worse than our base CNN model, which we hypothesize is due to the ResNet50 model being trained on the ImageNet dataset. Since this dataset consists of real-life, three-dimensional images, it does not seem to effectively learn the patterns of two-dimensional drawings prevalent in anime posters. The same concepts apply for semi-supervised learning, where complex and irrelevant patterns may have been learned. However, we did achieve the lowest Hamming Loss with our AutoKeras model, which means that the parameters set forth by AutoML performed slightly better than our base model. Therefore, a future exploration would be utilizing more AutoML techniques, such as H2O, to see if it can significantly outperform our base 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.

[8] Jin, Haifeng; Song, Qingquan; Hu, Xia. (2019). Auto-Keras: An Efficient Neural Architecture Search System. In \textit{arXiv preprint arXiv:1806.10282}.