Using Support Vector Machines and Neural Networks for image classification in Geek Culture

1 Introduction

Image classification is a task of giving an image a label assignment to identify which one category it belongs to among several categories. An example could be to classify whether there is presence of a cancerous tumor in an image produced by an MRI scan. One domain which has not seen much ML application is geek culture. McCain et al. (2015, p. 1) defined geek culture as "a subculture of enthusiasts that is traditionally associated with obscure media (Japanese animation, science fiction, video games, etc.)." However, these days it is widely believed that geek culture is no longer obscure but rather widespread. To a non-follower, there is hardly any difference between media within geek culture. The aim of this project is to facilitate a soon-to-be "geek" journey into three particular geek media: Japanese animation (referred to as anime in this report), Western cartoon and comics (referred to as comics in this report), and video games, particularly Pokemon-themed games (referred to as pokemon in this report). Because these media are quite similar to each other (e.g., similar presentation), distinguishing them can prove a formidable task. This is where ML could help since clearly a machine that can perform this task accurately would prove beneficial, especially for those already part of the subculture and wish to introduce their hobby to others.

This report presents and compares three different machine learning methods for assigning an image with one of the following labels: anime, comics, or pokemon. The structure of this report is as follows. In section 2, the problem of identifying whether an image belongs to one of the three categories is given a detailed formulation. section 3 gives a description of the dataset used in this project, the ML models, and the training process. In section 4, the performance of each model is assessed; model selection is performed. The last section provides a summary of the main discoveries, potential improvements in both the ML method and the training process are identified, and potential for future work.

2 Problem Formulation

The ML problem of this project is classifying whether an image is from anime, comics or pokemon. To be more specific, an image is classified as "anime" if it looks like a scene or character from Japanese "otaku" culture; "comics" if it resembles a scene or character from Western cartoon/comic "geek" culture; or "pokemon" if the image has a pokemon or a scene containing pokemon. Even though the Pokemon category may sound narrow compared to the other two categories, a running joke on the Internet is to identify whether something is Pokemon or a highly complicated concept[1]. Making Pokemon a whole category increases the difficulty of the classification task while making the problem more interesting. In this project, each datapoint is an image. The features of each datapoint are the raw intensities of each pixel of an image. Depending on the color mode of the image, each pixel is characterized by some number of values (one value for grayscale images, three values for RGB images). These are known as color channels. The number of features of an image is a product of its height, weight, and the number of color channels.

3 Method

3.1 The Data

The dataset used in this project is a collection of images of anime characters, cartoon or comic charaters, and images related to or containing pokemon. The images were gathered from various independent sources on Kaggle. Links to

[1]As an example: Is it Pokemon or Big Data?

each of those sources will be given in the references section. Anime, comics, and pokemon classes are indexed by numbers 0, 1, and 2 respectively.

There are a total of 391530 images in the aggregated dataset. The original images have different widths W, heights H, and number of color channels C from each other, and they all were resized so that all images have dimensions of 64×64 pixels, the same size as the smallest images, as it is generally recommended to downsize rather than upsize images because upsizing is just the algorithm guessing the new pixels. Additionally, every image was converted to RGB mode. Each image is a tensor (a multidimensional matrix) of dimension $W \times H \times C$ of pixel intensities. Since each pixel has three color channels (RGB), an image is characterized by $64 \times 64 \times 3 = 12288$ numbers, or features.

3.2 The Models and the Loss functions

This project demonstrates three algorithms: linear multi-class support vector machine (SVM), multilayer perceptron (MLP), and Residual Neural Network (ResNet). All of them, as well as the loss functions, were implemented, using the `Python` library `PyTorch`. The loss functions mentioned below were chosen because they are available in the aforementioned library, and because they are conventional for the models they were used on.

3.2.1 Support vector machine

The SVM was chosen as the first algorithm due to its interpretability and simplicity in classification tasks while serving as a good baseline model. Jung (2021, pp. 62–63) gives the details of the two-class SVM. In simple terms, a linear SVM is a linear model using the same hypothesis space as linear regression or logistic regression, and the SVM tries to learn a hyperplane to linearly separate one class from all the others. A linear multi-class SVM learns C hyperplanes at the same time, where C is the number of classes:

$$\mathbf{h}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{C \times D}, \mathbf{h}, \mathbf{b} \in \mathbb{R}^C$ and $\mathbf{b}$ is a bias vector. In this project, the original image tensor is flattened into a vector $\mathbf{x} \in \mathbb{R}^D, D = W \times H \times C$. Each row of $\mathbf{W}$ represents a hyperplane. $\mathbf{h}$ can be thought of as the vector of class scores. The label prediction is given by the index of the highest score $\text{argmax}_h(\mathbf{h}(\mathbf{x}))$. The hypothesis space for multi-class SVMs is $\mathcal{H} = \{\mathbf{h} : \mathbf{h} : \mathbb{R}^D \to \mathbb{R}^C\}$, where $\mathbf{h}$ is a linear (vector-valued) function and C is the output dimension. This algorithm minimizes the multi-class Hinge loss:

$$\mathcal{L}((\mathbf{x}, y), \mathbf{h}^{\mathbf{W}}) \propto \sum_{j=0}^{C-1} \max(0, 1 - \mathbf{w}_y^T\mathbf{x} + \mathbf{w}_j^T\mathbf{x}) \tag{2}$$

where $\mathbf{w}_i$ is the ith hyperplane, $j \neq y$, zero-indexing is used, and y is the index of the correct class. Note that this version corresponds to what is used in `PyTorch` (Torch Contributors, 2019b). In concrete terms, the SVM model used in this project has an input dimension of 12288 and output dimension of 3, meaning a total of $12288 \times 3 + 3 = 36867$ parameters since there is one weight for each feature, there are three classes, and the bias vector $\mathbf{b} \in \mathbb{R}^3$.

3.2.2 Multilayer perceptron

A short introduction to artificial neural networks, or just neural nets (NN) could be found in Jung (2021, pp. 68–70). In short, an NN is a composition of different functions ("neurons"). Each neuron could be connected with other neurons, and each of these connections is characterized by a weight parameter and possibly a bias term. The output of a neuron is then fed into another neuron after being transformed with a nonlinear function called an activation function. These neurons are usually organized into layers. In the case of multilayer perceptrons (MLP), each layer can be thought of as a linear vector-valued function, where the output dimension is the number of neurons. The perceptrons are exactly the neurons, each modelling a linear function of its inputs. In an MLP, every neuron in a previous layer is connected with every neuron in the following layer. The hypothesis space for MLPs is $\mathcal{H} = \{\mathbf{h} : \mathbf{h} : \mathbb{R}^D \to \mathbb{R}^C\}$ where $\mathbf{h}$ is a highly non-linear (possibly vector-valued) function. In a classification context, $\mathbf{h}$ is the vector of class scores. Figure 1 shows the architecture of the MLP used in this project.

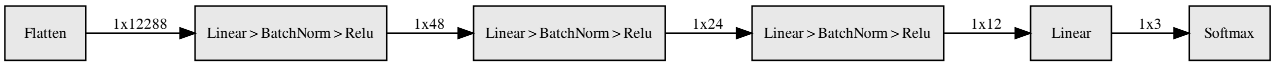
Flatten
1x12288
Linear > BatchNorm > Relu
1x48
Linear > BatchNorm > Relu
1x24
Linear > BatchNorm > Relu
1x12
Linear
1x3
Softmax

Figure 1: MLP architecture (original design).

The inputs are also flattened with the method described in section 3.2.1. After going through a linear layer, the immediate outputs get normalized to have the same mean and variance by a batch normalization function (BatchNorm), and then go through a rectifier activation (ReLU) function (Nair & Hinton, 2010). The numbers on the arrows show the input and output dimensions for each layer. The outputs of the final linear layer $\mathbf{h}$ are transformed by the softmax function to produce class probabilities. The label prediction is then the class with the highest probability. This MLP has about 590 000 parameters. The loss function for an MLP classifier is the cross-entropy loss (Torch Contributors, 2019a), where y is the index of the correct class:

$$\mathcal{L}((\mathbf{x}, y), \mathbf{h}) = -h_y(\mathbf{x}) + \log\left(\sum_{j=0}^{C-1} \exp(h_j(\mathbf{x}))\right) \tag{3}$$

3.2.3 Residual Neural Network

A residual neural network (ResNet), in the context of image-related ML tasks, is a special type of convolutional neural net (CNN), which is a class of NNs commonly used in image tasks. A CNN basically applies filters on raw image data (a $W \times H \times C$ tensor). Each of these filter (convolutional layer) learns a different characteristic (called feature in deep learning terminology) of the image, e.g., the edges of a face or hair color. The final layer then classifies the image based on these extracted features. He et al. (2016) give a formulation for ResNets. Figure 2 shows the architecture of ResNet-50 (50 layers) which was used in this project. Some of the terminology in the figure is from the same paper.

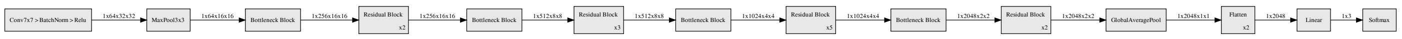

Figure 2: ResNet-50 architecture.

ResNets solve the problem of worse results for deeper nets than shallower nets by forcing some of the layers to learn a mapping $\mathbf{x} + \mathbf{f}(\mathbf{x})$ instead of the desired mapping $\mathbf{h}(\mathbf{x})$. He et al. (2016) empirically showed that an identity mapping $\mathbf{x} \to \mathbf{x}$ is easier to learn with the former formulation. The performance of the deeper net is at least the shallower net as the extra depth could just be identity mappings. The ResNet-50 model was slightly modified for this project: the final linear layer outputs a three-dimensional vector instead of a 1000-dimensional vector as there are only three classes. The loss function for this model is the same as the MLP and is given in Equation (3). This model also applies the softmax function to the final outputs. The ResNet-50 in this project has about 23.5 million parameters.

3.3 The Procedure

The data splitting is as follows. First, the dataset was manually split into a test set and set T. The size of the test set was about 30% of the original dataset. Next, set T was randomly divided (single split) into a train set and a validation set. k-fold CV was not used as the computational cost is too high (each training session took about three hours and used most of the resources of the computer). The size of the train set was 75% of the original train set. Half of the validation set was for tuning hyperarameters of each model (hyothesis space) and the other half for selecting between tuned models. Note that all models were trained and validated on the same train and validation sets. The test set was for evaluating the performance of the final selected model. All sets had a balanced class distribution, and for each class, all images roughly come from the same distribution. All model evaluation was done using the "accuracy" metric (average 0/1 error). This error is the proportion of misclassified examples to the number of examples. The 0/1 error was used to evaluate the models as this error is the common ground for all classification methods.

Table 1: Classification error (%) on train set and validation set for the three models.

	SVM	MLP	ResNet-50
Train error	27.62	15.35	**2.30**
Validation error	27.98	16.14	**2.81**

4 Results

Table 1 shows the 0/1 errors on the train set and validation set (the half for model selection) for all models. The validation errors were computed after tuning the hyperarameters of each model on the other half of the validation set. Validation errors were only a bit higher than their corresponding train errors, meaning that no overfitting is detected. These results were expected since overfitting was prevented in the training phase by tuning the hyperparameters and employing strong regularization techniques. Also, according to Jung (2021, p. 125), the small difference between train and validation errors suggest that the desired validation error was obtained, and not much could be done to improve the methods. In other words, the models could only achieve this much with the used train and validation sets, and with the optimization and regularization methods used when training and tuning hyperparameters.

Next, we notice that the SVM was the worst model while ResNet-50 was the best, no matter the train errors or validation errors. The ResNet-50 showed a great improvement over the SVM. Meanwhile, the MLP was right between the SVM and the ResNet-50. These observations suggest that NNs achieve better results than linear models such as the SVM when the inputs are raw pixel intensities. This is because a NN is composed of several functions, each learning different features of the image, whereas the linear SVM lacks expressive power due to its linear hypothesis space. Second, the ResNet-50 is the best model because ResNets work very well for image-related ML tasks as demonstrated by He et al. (2016). Nevertheless, the home-made MLP achieved an error rate below 20% (or an accuracy above 80%), which is already impressive considering the simplicity of the architecture. On this note, the SVM was also acceptable as its errors were all below 30%.

From the validation errors, we determined that the ResNet-50 was the best model. Next, we assessed the performance and generalizability of this model by testing it on the test set. The classification error of the model on the test set was 7.43%. In a classification context, the difference between the test error and the validation error was not too significant, and we can say that the model has good generalizability when given images similar to the original dataset. Note that the categories, anime, comics, and pokemon, are wide and difficult to define (especially the first two), limiting the performance of even the ResNet-50, which was trained on a very small subset of all possible images from anime and cartoon, in real-life production settings.

5 Conclusion

This report has presented three classification models on a dataset consisting of anime, comics, and pokemon images. The results suggest that neural networks might be better for end-to-end image classification (no initial feature engineering). Indeed, not having to worry about selecting the right features for our training is a main advantage of neural nets. However, while complex deep NNs such as a (previous) state-of-the-art ResNet can be applied to image tasks with overwhelming success (ResNet-50 error on the test set was 7.43%), we should use simpler models such as an MLP or even an SVM whenever possible as simpler models are more interpretable.

There is still much work to do. One thing to consider is improving the linear SVM. Instead of training it with raw image data, we could do feature engineering to reduce the dimensionality of the inputs and improve the learning efficiency of the model. Also, instead of a linear SVM, we could use kernel-based SVMs (see, e.g., Cristianini et al., 2000) which basically project the inputs into a space where linear separation is possible. Next is to use k-fold CV to get more unbiased validation errors as we might have "overfitted" the validation set. Another point is that in an image classification context, the current dataset is small and not diverse enough as e.g., it mainly consists of images with characters. This is why the trained models (and the final selected model) are expected to fail in production applications, e.g., when scenery and background images are given as inputs. Thus, we leave it to future work to collect more data, and use data with a wider distribution, i.e., more diverse images.

References

Cristianini, N., Shawe-Taylor, J., & others. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Jung, A. (2021). *Machine Learning: The Basics*. https://mlbook.cs.aalto.fi.

McCain, J., Gentile, B., & Campbell, W. K. (2015). A psychological exploration of engagement in geek culture. *PloS One*, *10*(11), e0142200.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. *Icml*.

Torch Contributors. (2019a). *CROSSENTROPYLOSS*. https://pytorch.org/docs/master/generated/torch.nn.CrossEntropyLoss.html.

Torch Contributors. (2019b). *MULTIMARGINLOSS*. https://pytorch.org/docs/master/generated/torch.nn.MultiMarginLoss.html.

References: Dataset Sources

Anime

http://www.nurs.or.jp/~nagadomi/animeface-character-dataset/

https://www.kaggle.com/soumikrakshit/anime-faces

https://www.kaggle.com/jahelsantiagoleon/female-anime-characters-anime-dataset

https://www.kaggle.com/profnote/pixiv-popular-illustrations

https://www.kaggle.com/splcher/animefacedataset

Cartoons and comics

https://www.kaggle.com/fivethirtyeight/fivethirtyeight-comic-characters-dataset

https://www.kaggle.com/jwiens/my-little-pony-friendship-is-magic-episode-data

https://www.kaggle.com/ymalov/simpsons

https://www.kaggle.com/alexattia/the-simpsons-characters-dataset

https://www.kaggle.com/mlwhiz/simpsons-main-characters

https://www.kaggle.com/vijayjoyz/tom-jerry-detection

Pokemon

https://www.kaggle.com/lantian773030/pokemonclassification

https://www.kaggle.com/thedagger/pokemon-generation-one

https://www.kaggle.com/vishalsubbiah/pokemon-images-and-types

https://www.kaggle.com/kvpratama/pokemon-images-dataset

https://www.kaggle.com/mornville/pokemonimagedataset

https://www.kaggle.com/brilja/pokemon-mugshots-from-super-mystery-dungeon

https://www.kaggle.com/aaronyin/oneshotpokemon

https://www.kaggle.com/subinium/various-pokemon-image-dataset