Final project for machine learning for Vision and Multimedia course: “Shazam Anime”

Author
Alessandro Bresciani
s305903@studenti.polito.it
ale.brex99@gmail.com

Author
Luca Filippetti
s303392@studenti.polito.it
filippetti99.lf@gmail.com

Abstract

Deep learning has made significant strides in accomplishing tasks such as classifying images and recognizing their contents, although the application of the latter in film and anime is still scarce. This paper aims to describe the use of such technology precisely to solve some of the problems related to this area.
During this project, we have attempted to implement a system capable of recognizing the title of an Anime from a single frame. Through the proposal of a Siamese-ResNet50 network, which combines a Siamese network structure with the well-known ResNet architecture, we evaluate the effect of its use by starting with a pre-training on ImageNet public data collection. We started by addressing the data collection problem, creating an adequate dataset despite the challenges of limited image availability. Next, we developed a solution based on using a Siamese neural network according to the strategy of few-shot learning coupled with the k-Nearest Neighbor (k-NN) algorithm. Compared with the use of a VGG-16, a ResNet50 shows greater skill and robustness in extracting features from different images.

1. Introduction

The explosion of multimedia content has led to an increasing demand for scene recognition and classification solutions from movies, TV series and anime. In this context, our project "ShazamAnime" aims to address a fascinating challenge: the recognition of Anime scenes from single images.
In the course of this project, we will adopt an approach based mainly on the few-shot learning paradigm [1], it differs from standard supervised learning in its goal, namely learn to learn: we try to recognize the similarity and difference between images and not to recognize the precise context of the images in the dataset and generalize it for the images in the test set. Our architecture is based on a Siamese neural network [2], which proves to be powerful in pattern recognition and development of complex representations. We will combine this approach with the use of a k-Nearest Neighbor (k-NN) algorithm to obtain high-precision results [3].
Next, we will provide a detailed overview of our project, outlining the challenges, importance, and approach taken. Despite the excellent prospects for development, possible limitations must also be considered: it is well known that extremely long training times and a large amount of data and model complexity are required to implement such a neural network. Our group, limited by the computational resources available and based on the content learned during the "Machine Learning for Vision and Multimedia" course, decided to contribute to the field of scene recognition of generic film (anime).

2. Dataset

Given the type of problem we sought to solve, it was necessary to conduct a completely customized data collection.
We made two datasets of anime images directly taken from a series of videos downloaded from the net (YouTube).
After completing an initial python script exploiting the NumPy library (data_set_extractor.py), it was possible to capture frames from all the downloaded videos at varying rates (25fps, 15fps, 10fps) to save the most significant ones from even the busiest videos leaving room in the code for any corrections or overwriting. Automatically the script saves the frames belonging to each specific class of cores, even from different videos, within each related folder, for a total of 16 classes (16 folders); at the same time, it randomly takes frames from each video to add them to the same classes, but in a separate folder designated for the network testing phase. The available classes are: A_Silent_Voice (203 images), Attack_on_Titans (208), Death_Note (206), Demon_slayer (208), Dragon_Ball (204), Fruits_Basket (201), Full_Metal_Alchemist_Brotherhood (205), Gintama (204), Hunter_X_Hunter (202), Jujutsu_Kaisen (210), My_Hero_Academia (208), Naruto (206), One_piece (205), One_Punch_Man (205), Re_ZERO (202), Steins;Gate (206).
Next, for each folder formed, it was essential to select and eliminate by hand the images that were too difficult for

network learning, i.e., excessively blurred and not very descriptive of the class in question, or too like each other to avoid possible overfitting. In addition, for the elimination of the Test images, it was found appropriate to delete those already present in the folder used for the realization of the train set, in this way the possibility of nullifying the testing operation was averted.
The entire Dataset obtained so far has been used for the classification phase entrusted to the final KNN, it corresponds to 200/204 images per class, for a total of 3280 images (from which derive those of the dataset for the Siamese) used for training and 714 of test images used only at the end for the k-NN, about 20%.
The format of the images is jpeg RGB, at 24-bit depth, 512x512 pixels and 96 dpi resolution.
For the creation of the Siamese-specific dataset, an initial script (make_siamese_dataset.py) was used to construct a CSV file containing all possible matches between the google drive paths of the images within the folders. So, for each path of an image, all the paths, of all the other images were matched, tying to each pair the label 0, in case it was the same class (the same source folder) and 1 otherwise. This allowed us to make combinations of images to train the Siamese network, which needs positive and negative cases explanatory of the similarity between animes.
The last script make_siamese_dataset_balanced.py, allowed us to balance the previously created CSV file. In fact, starting with all possible pairings resulted in a CSV file where pairs with negative labels, corresponding to different animes, greatly exceeded those with positive labels; therefore, the newly created CSV file aligns the number of negative cases with the number of positive ones to facilitate quick and correct training.
The dataset contained in the CSV file (siamese_ds_balanced) for the used Siamese Resnet50 network was loaded inside a dataset object of the tensorflow library. In total, the size of the dataset counts 1344976 pairs.
Then, after appropriate shuffling, it was processed at batch size of 128 to convert all the associated paths into actual image pairs. The final size of 256x256 pixels per image (type float 32) was selected because, given the resources available to us, it effectively allowed us to initiate experiments that would otherwise be impossible to conduct. The dataset was decomposed into one part for training, 1,210,496 pairs, and one for validation, or 134,498 pairs, 10 percent. Instead, the employed Test set corresponds to that derived from the first dataset, which contains 714 images, and it is used for the final k-NN.

3. Methods

3.1. Approach to the anime scenes recognition task

To tackle the task of recognizing anime scenes from images, we adopted an approach based on a combination of a Siamese neural network and a k-Nearest Neighbor algorithm (fig. 1). This choice was dictated by the nature of the problem, which required a model capable of learning the subtle differences between anime scenes without resorting to intensive training of a traditional classifier.
The approach is based on the idea of teaching the Siamese neural network to measure the similarity between two images. This learning process is done by comparing image representations extracted from a pre-trained neural network backbone on ImageNet. The Siamese neural network calculates the distance between the representations of the two images and learns to distinguish between similar and different images.
Next, we use the k-NN algorithm to classify the test images. The advantage of this approach lies in its scalability and its ability to handle large amounts of data without requiring additional training.

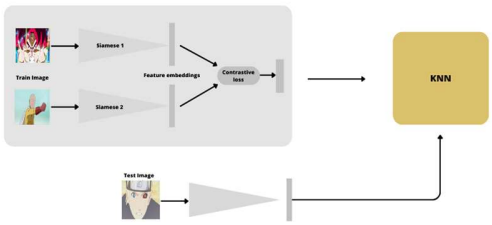

Figure 1. General outline of the approach (Siamese network and k-NN).

3.2. Siamese model architecture

The architecture of our Siamese neural network begins with the ResNet-50 backbone. The ResNet-50 model belongs to a family of artificial neural networks known as residual networks (Residual Networks). This network has been pre-trained on ImageNet, which means it can extract meaningful features from complex images.
The extracted features go through a Global Average Pooling and a Dense layer with 512 units, making the representations even more informative. We then use a Lambda custom layer to calculate the Euclidean distance between the features obtained from the two branches of the Siamese network, which is crucial for measuring the

similarity between images (fig. 2).

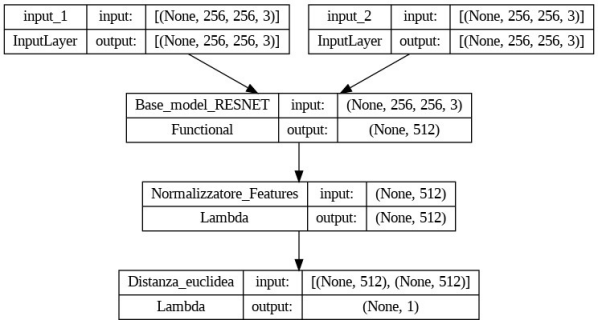

Figure 2. "Summary" of the Siamese network architecture used.

3.3. Loss function, mathematics and training of the Siamese network

Learning rates: 1.0/0.1 – 0.00006/0.00001 and adaptive lr
Optimization deployed: Adam, ema_momentum 0.99.
Loss function: Contrastive loss function
Batch size: 128 samples
Training images dimensions: 256x256
Base model: Resnet50, 175 layers

During the training of a Siamese network, two or more inputs (two in our case, being a one-shot learning) are encoded. The two branches of the Siamese network share weights and are trained simultaneously. The goal is to teach the network to map images into a feature vector representation space in such a way that similar samples end up closer, while dissimilar ones end up farther apart. While learning the network, it was decided to employ a function that calculates the Euclidean distance

$$d(i,j)=\sqrt{\sum_{k=1}^{n}(x_{ik}-y_{jk})^2}$$

between the pairs of training images by producing the result after transforming them into embeddings. In order to train the network, it was decided to exploit the so-called Contrastive Loss, the full formula of which is:

$$L_{\text{contrastive}}(Y,D)=\tfrac{1}{2}(1-Y)D^2+\tfrac{1}{2}Y\max(0,m-D)^2$$

it takes as input a pair of similar or dissimilar samples, moves similar samples closer together and dissimilar samples farther apart. Given a pair of images *(I1, I2)* and a label *Y* equal to 0 if the samples are similar and equal to 1 otherwise. To extract a low-dimensional representation of each sample, we use our F network that encodes the input images in a space where ***x1 = F(I1)***, ***x2 = F(I2)*** (fig. 3). During backpropagation, the network will try to minimize the contrastive Loss, so if the samples are similar (Y=0), the ***D***, i.e., the Euclidean distance between x1 and x2, is minimized, on the other hand, if the samples are dissimilar, (Y=1), the second term is minimized, i.e., their Euclidean distance will be maximized up to a certain selectable limit m.
Adam was chosen to be employed as the network optimizer. It belongs to the family of adaptive gradient methods that differentiate the learning rate for each parameter. At each time instant, Adam computes the moving average of the mean and variance of the gradients by exploiting momentum (a tool for increasing the rate). This choice was made because, based on teaching labs and references to Siamese network papers, it was determined to be the most adaptive and performant.

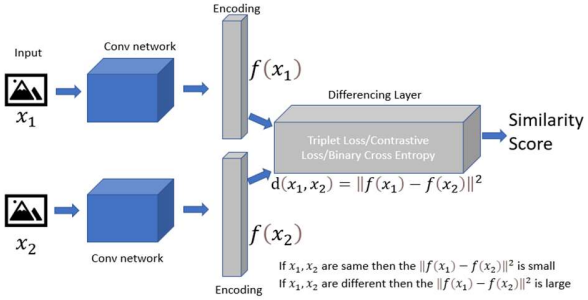

Figure 3. Full Siamese network model.

For all trainings attempted, complications arising from changing the number of units of the final Dense layer, added to extrapolate feature vectors once the entire training process was completed, were analyzed. It was noted, after several attempts, that the best compromise was to employ the 512-unit Dense layer. Unfortunately, the available resources were not sufficient to employ Dense layers of 2048, i.e., equal to the deep output from the Resnet (16, 16, 2048), which would have ensured better accuracy in the final testing phase.
In addition, it was decided to keep the starting dataset particularly rich in available samples, both in order to be able to avoid underfitting the network and to obtain better results in performance: an experiment was conducted by also exploiting the original, unbalanced dataset, which created an impossibility in continuing the training given the number of existing pairings. Another attempt was conducted by employing a reduction of the dataset for the Siamese network by half.
All experiments that considered an image size of 512x512 pixels failed; training was unable to begin with the resources available to us. During training, we operated the transfer learning (TL) strategy to improve Siamese network learning (with Resnet50). It consists of an initial "warm-up" phase, in which after constructing a base model (backbone), we eliminated the terminal layers (dense layers) of ResNet50, and then trained only the added layers of the Siamese network on our core scene

dataset, namely a global average pooling, a Dense layer, a feature normalizer, and a Lambda layer for Euclidean distance computation with a batch size of (128, 256, 256, 3). During this phase, it was possible to train for up to 2 total epochs, employing the strategy, at each total epoch, of one step per epoch equal to 256.
A few configurations were tested to acquire the best performance.
The configurations used started from a batch size of (256, 256, 256, 3), with which we found great difficulty in advancing the training due to the large size of the latter. Therefore, in subsequent experiments we reduced the batch size to (128, 256, 256, 3) and trained for a total of 37 epochs (steps per epochs equal to 256) by implementing saves every 128 steps via a checkpoint callback so that in case of interruptions, we have the possibility to reload the model and its state again. During the fine-tuning phase of the network, the first experiments were performed exploiting the entire network completely "unfreeze", through a batch size of 32 , 128 steps per epochs and 296 epochs, with a learning rate of 0.1: the experiment was a failure and due to the limited resources available (saturation of system RAM and GPU RAM), it was possible to train only up to the eighth epoch, achieving a Loss and valuation Loss of 0.22.
Subsequent attempts were based on a modification of the steps per epochs and learning rate (reduced to 0.0001) but failed in the same way.
So, it was decided to exploit partial fine-tuning: the first 140 out of 175 layers of ResNet50 were made untrainable, operating on only 65% of the total parameters.
Again, several configurations were experimented with by changing batch size and learning rate. In many cases, it was decided to employ an exponential decay of 0.1 every 8 epochs (out of the total 37) and a fixed decay (using a callback) to decrease, again by the same rate, the learning rate in case the optimizer encountered plateaus. This was accomplished based on various research, and it was found that it would greatly help the optimizer.
The experiments were conducted by setting the batch normalization layers in inference mode. It is a good practice for when performing fine tuning operations. By setting the layers to freeze during the warmup phase, the Batch Normalization layers will be run in inference mode and their statistics will not be updated so when they are subsequently unfreezed, it is necessary to maintain the inference mode since otherwise untrained weight updates could destroy what the model has learned due to the presence of two different distributions in the layers.

In this version, the warmup phase was conducted with batch sizes of 128, steps per epochs of 256, for a total of 37 epochs and an exponential decay learning rate of 0.1 starting with 0.1 as the initial value and ending at 0.00001. Instead, an exponential decay learning rate of 0.1 starting from 0.0001 was used for fine tuning. We obtained a loss of 0.236 for warmup and 0.221 for fine tuning (fig. 4).

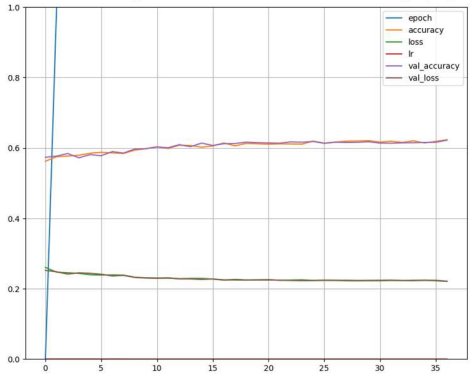

Figure 4. Graph showing the trend of parameters.

A second experiment was performed with the same parameters, but attempting to exploit a lr equal to 1.0, decaying every 15 epochs to 0.1/0.001 and every 2 consecutive epochs of the same valuation loss.
The same hyperparameters were applied in the Fine-tuning phase.

From these fine-tuning workouts, it was noticed that the loss diverged if the starting lr (learning rate) was high, probably this issue shows us that the model is unstable and not completely fit to solve the task.

It was also decided to compare the previous experiments with a new version of the architecture, that is, trying to place, against indications, the bath normalization layers in training conditions during warmup and fine tuning.
We concluded the warmup phase by training the network with a lr of 0.00006 until the 20th epoch, and then ended the epochs at lr 0.00001: the result was disappointing, the Loss did not drop below 0.23, probably due to the limitation of model complexity at this stage.
Further final experiments would be possible to conduct by modifying the optimizer. Through careful research, another optimizer besides Adam, which is efficient in Siamese networks, was discovered, namely Nadam. Otherwise, an SGD optimizer would not give optimal results.

3.4. Model surgery and classification

After training the Siamese network, we performed a "model surgery" to extract one of the two branches of the Siamese network, using it as a feature extractor. (fig. 5)
The k-Nearest Neighbor (k-NN) is a nonparametric supervised learning classification algorithm that uses proximity to make predictions. Given a vector to label, one looks for the 'k' vectors in the train set that are "closest" to the test vector. We then assign the most frequent label among the 'k' labels. To calculate the

closeness of the elements, we used a Euclidean distance: d: ={(x1-y1)^2+(x2-y2)^2}^1/2.
The choice to use the k-NN algorithm was dictated by several considerations: first, the number of classes or headings in our context is relatively large, especially as a function of what would be a truly feasible system, which would have made training a dense classifier a complex challenge in terms of the resources required. In a field where new anime titles may emerge frequently, the ability of k-NN to adapt quickly to new classes without requiring a full training procedure makes it a more pragmatic choice. Another important consideration was the criticality of inference time, for while k-NN may be more weighted than a dense classifier, this difference is negligible for applications where the main priority is classification precision.

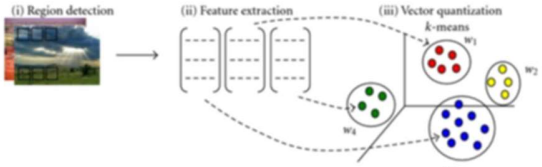

Figure 5. High-level schematic of a feature extraction e k-NN.

3.5. Performance and metrics evaluation

To evaluate effectiveness, we used several evaluation metrics, among them one of the most informative was the confusion matrix, as well as a simple accuracy value. The confusion matrix compares, for each class, the values predicted by the model with the actual values existing in the data. Specifically, the rows of the matrix represent the model predictions and the columns represent the actual data. Indeed, thanks to the matrix it is possible to understand which classes the model confuses the most.
To obtain a detailed visualization of the extracted features, we adopted the t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm. The t-SNE is a widely used dimension reduction technique to visualize data in a lower dimensional space while preserving as much as possible the structure of the original data, in our case three dimensions. The use of t-SNE provided us with a clear representation of the features extracted from the Siamese network and facilitated visual analysis of the relationships among the images in the dataset, in particular, it helped us to verify whether the features were learned effectively (fig. 6) and whether the approach was able to create distinct representations for the different core classes.

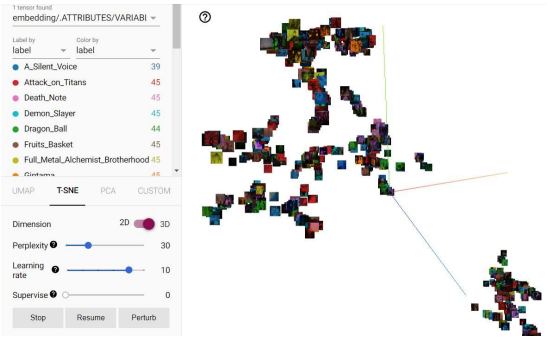

Figure 6. Screen of the T-sne clusters of the "ShazamAnime," (the images have a coloring according to their own label).

4. Experiments and results

To evaluate the results of our network, we decided to compare the trained model with a baseline ResNet50 network not trained on our dataset, but only pre-trained on ImageNet. The results are unfortunately not satisfactory. In fact, as can be seen from the confusion matrix (fig.7) and the accuracy of 0.08 (fig. 8) the network is unable to distinguish between classes.

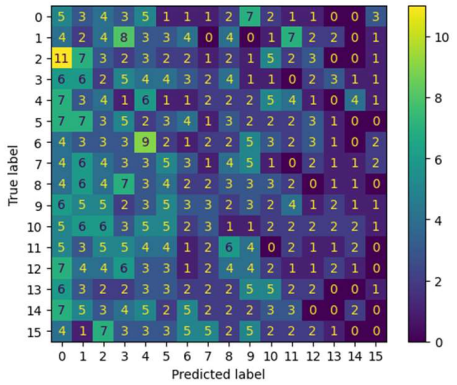

Figure 7. Confusion matrix results of "ShazamAnime".

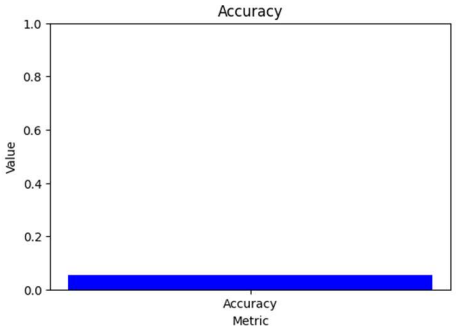

Figure 8. Accuracy of "ShazamAnime".

An additional experiment was also carried out, as a pure curiosity, to perform the final image classification by substituting a Dense layer equipped with a softmax activation function for the k-NN. Once one of the branches of the Siamese was cut off, it was necessary to insert the new Dense layer. Training was conducted at batch size 128 and fixed learning rate of 0.01 (one loss: 2.73 to 2.5; val_loss: 2.7 to 2.53; accuracy: 0.10 to 0.21) for the first 10 epochs. After that, further training was carried out: 24 epochs at fixed learning rate (0.001) and 38 epochs at variable lr (exponential decay) starting from 0.001 with steps every 8 epochs up to 0.00001.

Epoch 10	Lr = 0.01	Loss.i = 2,73	Loss.f = 2.5	Acc=0.21
Epoch 74	Lr: 0.001-0.000001	Loss.i = 2.47	Loss.f = 2.46	Acc = 0.24
From scratch: Epoch 38(1)	Lr: 0.1 – 0.00001	Loss.i = 2.7	Loss.f = 2.26	Acc = 0.32
From scratch: Epoch 128 (2)	Lr: 0.01	Loss.i = 2.04	Loss.f = 1.99	Acc = 0.39
From scratch: Epoch 256 (3)	Lr: 0.1 – 0.01 Batch size 256	Loss.i = 2.7	Loss.f = (1.84) 1.73	Acc = (0.47) 0.48

After training that classifier, the 2 best results obtained compared with those of k-NN were analyzed:

X	k-NN	Dense classif_1	Dense classif_3
accuracy	0.08	0.074	0.061
loss	X	2.26	1.73
Valuation_loss	X	2.32	1.87

The confusion matrix gets about the same results as the k-NN. In each case, although unsuccessful, the k-NN gives the best result.

5. Conclusions

Training a model capable of recognizing the title of an anime from static images was a challenging task. The very nature of the task, which requires understanding image content in the absence of context, necessitated a large and diverse dataset. In addition, the limited computational resources of Google Colab posed a challenge in training a complex neural network.

Importantly, model accuracy and performance could be improved.

It would be logical to employ feature vectors at least as large as the output size of the employed backbone (2048), as this would increase the accuracy of the features. In addition, the size of the images has been greatly reduced, good results could be obtained by employing images with a high resolution, such as 1024x1024, so that the network could optimally learn the more specific (recognition of corners, edges, etc.) and more generic (eyes, faces, limbs, etc.) features of each anime representation.

Obvious improvements would be made by completely unfreezing the entire ResNet50 network and training all the parameters during the fine-tuning phase. Such an operation would result in a fully deep network, which would learn higher-complexity elements, while solving the problem of vanishing and exploding gradients and weight variability would require employing regularization of the latter.

Having started with an extremely dense dataset of samples, it was not computationally possible to implement a data augmentation technique; it would be recommended for future experiments to reduce possible overfitting.

6. References

[1] Yinbo Chen, Zhuang Liu, Huijuan Xu, Trevor Darrell, Xiaolong Wang <<*Meta-Baseline: Exploring Simple Meta-Learning for Few-Shot Learning*>> arXiv:2003.04390v4, 2021.

[2] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov <<*Siamese Neural Networks for One-shot Image Recognition*>> Department of Computer Science, University of Toronto. Toronto, Ontario, Canada, 2015.

[3] Benjamin Villard, Yuichi Mori, Masashi Misawa, Shin-ei Kudo, Hayato Itoh, Masahiro Oda, Kensaku Mori <<*Colorectal Polyp Size Classification Using a Siamese Network*>>, 2019.

[4] github: One-shot Siamese Neural Network: https://github.com/nevoit/Siamese-Neural-Networks-for-One-shot-Image-Recognition.