IN3062 - INTRODUCTION TO ARTIFICIAL INTELLIGENCE

Coursework

Sherwin Domingo F. – acwf556
Wadha Alhamdan – acts796

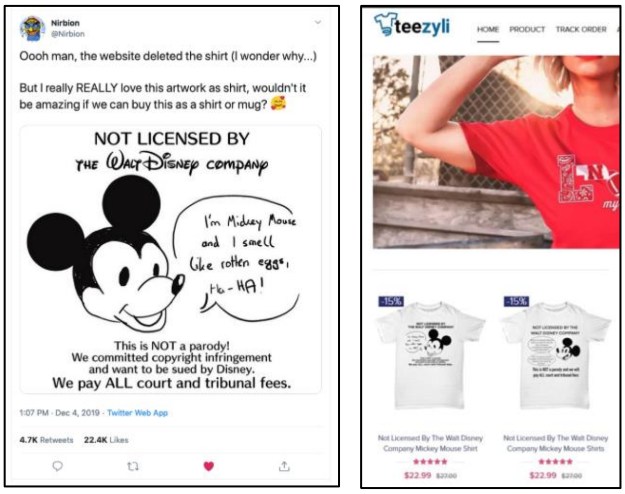
Table of Contents

Introduction

As online shopping is slowly dominating the e-commerce market due to the benefits that are presented to the buyers, such as convenience, accessibility, privacy for discreet purchases, etc; organisations are finding it more difficult to ensure that their products and intellectual properties are not being plagiarised and/or sold without their consent. As a result, due to the scale of this sector in the market, it is less likely for these unlawful online businesses to be sued for violating their copyrights and patents. For example, in 2014 Disney started proactively searching through numerous online sites to crack down on all unlicensed vendors selling unlicensed Disney merchandise (Doctor Disney, 2014). However, it would be more difficult for independent artists and freelancers to keep track of their products not being plagiarised.

Problem domain

For a long time now, independent artists in social media have come across websites selling their artworks as t-shirts, mugs and other merch without their permission or compensation. They had a theory where they believed that there were bots constantly lurking through their social media activity while analysing the followers' reaction to their posts. This theory was then tested by @Hannahdouken on Twitter on Dec. 3 by encouraging the followers to reply to the post by saying 'I want this on a shirt' to test whether the bots would pick up the replies and upload the image on the website. Sure enough, bots picked up the responses and proceeded to then upload the image to websites and sell the 'fake' artwork as merch (Polygon, 2019). The products were taken down right away, but another user on Twitter tested the theory as well and the bots picked it up too, refer to Figure 1.



Nirbion
@Nirbion

Oooh man, the website deleted the shirt (I wonder why...)

But I really REALLY love this artwork as shirt, wouldn't it be amazing if we can buy this as a shirt or mug? 🥰

NOT LICENSED BY
THE WALT DISNEY COMPANY

I'm Mickey Mouse and I smell like rotten eggs, Ha-HA!

This is NOT a parody!
We committed copyright infringement and want to be sued by Disney.
We pay ALL court and tribunal fees.

1:07 PM · Dec 4, 2019 · Twitter Web App

4.7K Retweets 22.4K Likes

teezyli
HOME PRODUCT TRACK ORDER
-15%
-15%
Not Licensed By The Walt Disney Company Mickey Mouse Shirt
$22.99 $27.00
Not Licensed By The Walt Disney Company Mickey Mouse Shirts
$22.99 $27.00

Figure 1. Screenshot of the tweet sent by a user testing the 'bot theory' and its result.

Dataset

The original dataset titled "complete-pokemon-image-dataset" previously can be found on Kaggle, however not anymore, here is a link to the original zip. it contained more than 900 classes each containing around 20-40 images. For this project, we picked some *Legendary* Pokémon's as our dataset as these are more likely to be uploaded on websites and sell their design as merch since the majority of Pokémon fans believe that they are the strongest ones. Therefore, we cut down our dataset to a set of images of 11 Pokémon's each set containing 40 images thus making it 440 images in total. However, at the beginning the dataset was unbalanced as some labels contained more images than others. This issue has been solved using the techniques described in the 'Missing data' section.

Classification

All of our models follow a supervised learning method. The goal is for the machine to successfully classify each image to the correct Pokémon. Each model first uses an input function to get the training set and testing set from the directory. Then, we apply Principal Component Analysis to the dataset to produce a low-dimensional representation of the dataset thus making it easier to process and achieve a much better performance as our models tend to work better in a low-dimensional space. Finally, each model's algorithm is applied to the dataset to train the machine and then expect its output predictions.

Missing data

Some Pokémon's, or labels, had more images than others thus making the data bias as the machine would learn from some labels more than others. When we first downloaded the dataset the label with the most images had 38 while the rest of labels struggled to even hit the 30 mark. As a result, when the machine first learned the dataset without us completing it, there was a higher accuracy of prediction on the label with the most images.

Therefore, we decided round up the images of each label to 40 to achieve unbiased results. To do this, we re-searched and downloaded Pokémon images and added them to our dataset. Moreover, we have used Photoshop to edit some of the images to ensure that they all had the same dimensions and the same type (.jpg) we also ensured that none of them were duplicates.

Techniques used

To understand our dataset, we have decided to use colour histograms for the images. Most people identify Pokémon based on their colour first, and their figure second. Therefore, we wanted to investigate how the computer see the images of Pokémon. The following table shows the histogram for some images:

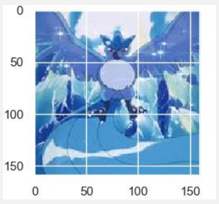
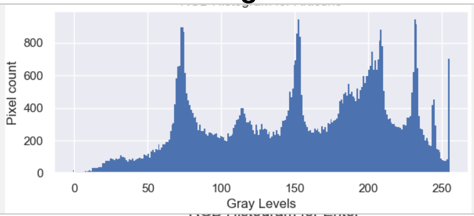
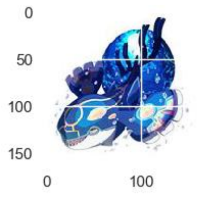
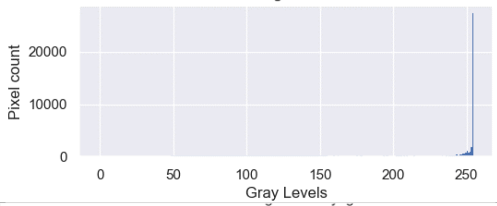
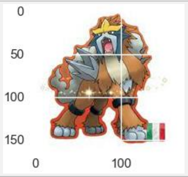
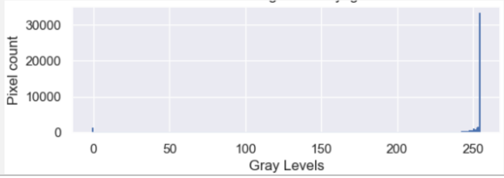
Class	Original Image	Histogram
Articuno		
Kyogre		
Entei		

Table 2. Showcasing histogram for some classes.

As you can see from Table 2., the last 2 histograms are skewed to the right with a big difference in pixel counts. This is due to some of our images inside the dataset having a plain coloured background (mostly white and black). Because of this the pixel count for those colours become extremely larger than the pixel count for the actual colours of the Pokémon. As a result, the histogram becomes bias which we believe it could affect the results for some of the labels. For example, the background colour of an image from Zapdos class (a yellow Pokémon) could be blue. Since the pixel count for the background is larger than the pixel count of the Pokémon, this could confuse the machine thus making it predict that the image belongs to Articuno or Kyogre class (both blue Pokémon).

In the future, to fix this, we could ensure that every Pokémon is in front of a black background and write up a code to make the machine ignore the black pixels to achieve a higher prediction accuracy.

Models used

For this project we have decided to use 4 models:

- Support Vector Machine (SVM)
- K-Nearest Neighbours (KNN)
- Random Forest (RF)
- Convolutional Neural Network (CNN)

All models, but the CNN, have been running with and without PCA to see whether producing a low-dimensional representation of the dataset was more effective and more accurate than without it. These results will be reflected in the 'Results' section of this document. Additionally, we have tested each model with 3 different types of data (All Features, Just Pixels, Extracted Features). These are explained in the next section.

Finally, when testing our models with PCA, we have produced accuracy plots to see how well they would perform given a different number of components. Please refer to Figure 3. inside the Results section for these plots.

Support Vector Machines
SVM model can be easily converted into a non-linear classifier which we predicted that it will work well with our dataset given that we had 11 classes that had to be separated. This model works well in low dimension space than other models thus achieving higher accuracy results when PCA is applied (52%). However, without PCA, the other models performed far better than SVM as it struggled to achieve more than 5%. This is reflected on our results when applying PCA to our dataset compared to the other models (except for CNN). However, this model is not very efficient when the dataset is big as it requires a large amount of time to process (Towards Data Science, 2018). This could be a problem in the future when in the real world as the complete dataset of all Pokémon's would have more than 800 labels each containing 40 or more images.

K-Nearest Neighbours & Random Forest
Unlike SVM, k-NN would classify data based on the similarity of its neighbours making this model a lazy algorithm. This means that it does not use the training data points to do any generalisation thus making the training phase faster. This would be helpful in the future as the Pokémon dataset will be more extent thus saving time on the training phase. However, as this algorithm stores all of the training data. If we apply this method to a very large dataset, such as the Pokémon one on a larger scale, the prediction stage would be very slow as it would require a lot of memory and processing power.

On the other hand, when Random Forest, a special form of decision tree algorithm, is used with few features, it could help us reduce processing power on our dataset, as the computational cost of training is low, and would still get us good results, refer to 'Results' section. In contrast, it will reach a point where more samples will not improve the accuracy. Consequentially, in the real world where we use a large dataset, it will reach a point where no matter how much more this model is trained, the accuracy won't change thus making it unreliable if the model is not accurate enough.

Our aim was to investigate how these model's results differ from them SVM's. We expected SVM to perform much better than these two models given how SVM is portrayed on paper. However, as you can see in the Results section, both models performed a lot better without PCA. Moreover, when PCA is included, k-NN and Random Forest tend to perform almost as good as SVM with a fewer number of components.

Convolutional Neural Network
Are a specialised type of Neural network that excels in processing imagery and video data, it consists of a convolution network connected to a neural network (Goodfellow et al, 2016). The convolutional network performs a mathematical operation called convolutional which can efficiently extract different representations in images, which is then put through the neural network. Both sections have weights that are trained to both better extract representation and make predictions. We predicted the CNN to outperform the previous classifiers, considering its remarkable representation extraction process, which it has, achieving a remarkable 83% accuracy.

Encoding the input variables
The data has been put in a specific folder structure such as, "images/class_name/image.jpg" using this we have made a simple function that extracts each image and its corresponding label into the variables "X" and "y". the images are kept in their RGB format and converted into image arrays, while the string labels are one-hot codded using sklearn's "LabelBinarizer".

Feature engineering is the process of extracting new features from the feature space using domain knowledge to increase the amount of data we have (Wikipedia, 2019). We wanted to test out different types of data with our models, so we made 3 different data types. "just pixels" refers to the flattened image arrays, "extracted features" are just the extracted features, and "all features" is the mix of the previous two. The extracted features consist of performing the following statistical operations on all the pixels and on each RGB channel, which are finding the mean, median, and standard deviation. Thus, we have 12 new features we were able to extract using our knowledge of RGB channels and our previous analysis of grey levels. We hope this slight increase in feature space helps the models perform better.

Accuracy Evaluation Criteria
The models are evaluated using different methods that emphasize different aspects of the model. All models are run through sklearn's "classification_report" method, which produces a report that calculates the precision, recall, f1-score and support of each class. It also calculates the accuracy of the model, the macro average and the weighted average of each of the precision, recall and f1-score. Additionally, we produce a

confusion matrix plot using Seaborn’s library, specifically the method “heatmap”, which helps visualize the classification report. Moreover, we created an accuracy over PCA component plot for each of the data types, with and without PCA for the models RF, KNN, and SVM. Furthermore, for the RF model we produced an accuracy over number of trees to find the most optimal tree count n. Lastly, for the CNN model we created 2 more plots for it, a loss over epoch plot, and an accuracy over epoch plot, these display how the training and validation sets perform and help with hyper-tuning.

Results

For the classifiers RF, KNN and SVM, three different forms of the data were run on them (all features, just pixels, only extracted features) with different PCA configurations. In contrast, the CNN only trained on the just pixel data without PCA configuration since it’s able to extract its own representation very efficiently. The testing set will be 25% of the dataset so around 110 images, additionally CNN will have a validation set which will be 20% of the training set. Before going to find the best accuracies, we needed to figure out the most optimal PCA n component and most optimal number of trees for the RF model. Please refer to Figure 3. for some examples of the produced accuracy over PCA n component plots for the All Features data type and 1 example of the accuracy over number of trees for the RF model.

We can see from Figure 3. that different PCA configuration greatly affect the accuracy of the model. All the models behave negatively when the PCA is lower than 10, so we can conclude that 10 dimensions and less is not enough to capture the data well. RF and KNN react negatively on higher dimensions, when choosing the correct PCA configuration they almost always perform better. As for the RF specific plot, the most optimal number of trees is around 130, more than this and the accuracy gets very unstable. These plots have helped us narrow the range of PCA and number of trees configuration to test.

Different PCA configurations were used in the testing, however only the configurations that achieved the highest average accuracy were put in Table 4. please refer to it for an overview of all accuracies achieved.

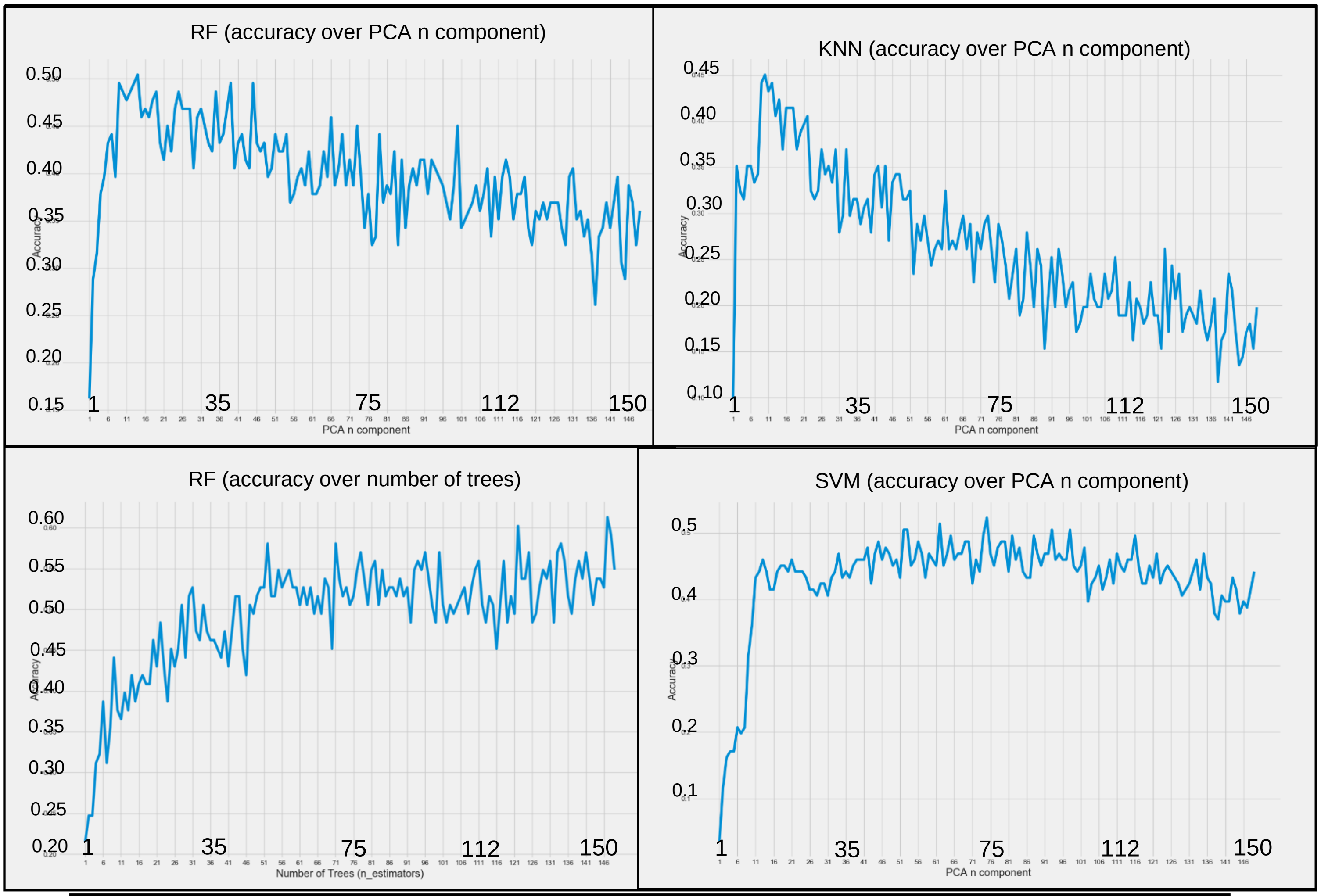


Figure 3. Accuracy over number of trees plot for RF and Accuracy over PCA n components for the classifiers, RF, KNN, and SVM on the data type: All Features. The x-axis ranges from 1 to 150.

Accuracy on test set	**Type of Data**		
Classifiers	All Features (pixels + extracted data)	Just Pixels	Just Extracted Data
Random Forest (RF)	No PCA = 41% With PCA(14) = 50%	No PCA = 40% With PCA(15) = 49%	No PCA = 20% With PCA(10) = 15%
K-Nearest Neighbour (KNN)	No PCA = 30% With PCA(15) = 45%	No PCA = 30% With PCA(10) = 45%	**No PCA = 25%** With PCA(10) = 16%
Support Vector Machine (SVM)	No PCA = 5% **With PCA(75) = 52%**	No PCA = 5% With PCA(75) = 52%	No PCA = 19% With PCA(10) = 14%
Convolutional Neural Network (CNN)	No applicable	**No PCA = 83%**	Not applicable

Table 4. Overview of all accuracies achieved across all models and data types using different configurations of PCA.

Overall, all models behave positively with the addition of the correct PCA configuration, this behaviour has been previously observed in a study comparing different text classifiers with and without PCA (Taloba, Eisa, Ismail, 2018)

Support Vector Machines

No PCA-PCA	*All Features PCA n comp = 75*			*Just Pixels PCA n comp = 75*			*Extracted Features PCA n comp = 10*		
Class	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Articuno	0% - 29%	0% - 31%	0% - 30%	0% - 29%	0% - 31%	0% - 30%	17% - 0%	15% - 0%	16% - 0%
Entei	0% - 62%	0% - 50%	0% - 56%	0% - 62%	0% - 50%	0% - 56%	40% - 43%	20% - 30%	27% - 35%
Groudon	0% - 80%	0% - 67%	0% - 73%	0% - 80%	0% - 67%	0% - 73%	19% - 21%	25% - 25%	21% - 23%
Kyogre	0% - 46%	0% - 60%	0% - 52%	0% - 46%	0% - 60%	0% - 52%	18% - 17%	40% - 10%	25% - 12%
Latias	0% - 82%	0% - 64%	0% - 72%	0% - 82%	0% - 64%	0% - 72%	29% - 8%	14% - 7%	19% - 8%
Latios	4% - 10%	100% - 25%	7% - 14%	4% - 10%	100% - 25%	7% - 14%	5% - 0%	25% - 0%	9% - 0%
Moltres	100% -50%	12% - 62%	22% - 56%	100% - 50%	12% - 62%	22% - 56%	33% - 11%	25% - 12%	29% - 12%
Raikou	0% - 33%	0% - 80%	0% - 47%	0% - 33%	0% - 80%	0% - 47%	0% - 17%	0% - 20%	0% - 18%
Rayquaza	0% - 67%	0% - 46%	0% - 55%	0% - 67%	0% - 46%	0% - 55%	33% - 18%	15% - 23%	21% - 20%
Suicune	0% - 50%	0% - 40%	0% - 44%	0% - 50%	0% - 40%	0% - 44%	43% - 7%	30% - 10%	35% - 8%
Zapdos	100% - 100%	8% - 50%	15% - 67%	100% - 100%	8% - 50%	15% - 67%	0% - 10%	0% - 8%	0% - 9%
Overall Accuracy	All Features			Just Pixels			Just Extracted Data		
	No PCA = 5% With PCA (14) = 52%			No PCA = 5% With PCA (15) = 52%			No PCA = 20% With PCA (10) = 15%		

Table 5. SVM's classification report. Without PCA (blue colour) and with PCA (red colour)

When compared to the RF and K-NN models, our SVM model has reached the highest accuracy when applying PCA and it's also the model with the highest accuracy except for CNN. Moreover, different n components wouldn't drastically change the accuracy values, whereas RF and K-NN models would have a decrease in accuracy as the n components increases, refer to Figure 3. The parameters used in this model are mostly default except for the 'Regularization parameter' which is set to 5.0 instead of 1.0.

As we can see from Table 5., this model performs much better with a low-dimensional representation of the dataset as the model ran with PCA has drastically improved its accuracy. Interestingly enough, if we take a closer look to the table, the accuracies for different types of data (all features and just pixels) have the exact same results. We can conclude from these results that SVM has indeed a good separation as the margin is generated consistently. However, a downside we have observed is that no matter how many times we train this model, it will always give us the same results. On the other hand, we can observe from the Just Extracted Data column that SVM does not perform well when only 10 features from the data are extracted. However, when PCA is not used, it performed better than in other data types.

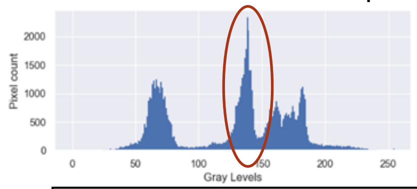


Figure 6. Latios' histogram

The class type *Latios* and *Articuno* seemed to be the most misclassified overall and not being classified at all when using Just Extracted Features data. If we take a look at the histogram for *Latios* (Figure 6), we can identify a great domination by blue pixels. Since these 2 Pokémon have similar colours (White & Blue), that could be the cause for this misclassification issue.

K-Nearest Neighbour

No PCA- PCA	*All Features PCA n comp = 10*			*Just Pixels PCA n comp = 10*			*Extracted Features PCA n comp = 10*		
Class	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Articuno	29% - 29%	38% - 31%	33% - 30%	29% - 29%	38% - 31%	33% - 30%	24% - 16%	31% - 38%	27% - 22%
Entei	56% - 50%	50% - 50%	53% - 50%	56% - 50%	50% - 50%	53% - 50%	40% - 43%	40% - 30%	40% - 35%
Groudon	46% - 50%	50% - 50%	48% - 50%	46% - 50%	50% - 50%	48% - 50%	25% - 14%	33% - 17%	29% - 15%
Kyogre	33% - 55%	20% - 60%	25% - 57%	33% - 55%	20% - 60%	25% - 57%	21% - 15%	40% - 20%	28% - 17%
Latias	50% - 50%	29% - 43%	36% - 46%	50% - 50%	29% - 43%	36% - 46%	50% - 33%	29% - 7%	36% - 12%
Latios	10% - 9%	50% - 25%	16% - 13%	10% - 9%	50% - 25%	16% - 13%	0% - 0%	0% - 0%	0% - 0%
Moltres	33% - 43%	12% - 75%	18% - 55%	33% - 43%	12% - 75%	18% - 55%	38% - 9%	38% - 12%	38% - 11%
Raikou	6% - 33%	20% - 20%	9% - 25%	6% - 33%	20% - 20%	9% - 25%	12% - 20%	20% - 40%	15% - 27%
Rayquaza	43% - 80%	23% - 31%	30% - 44%	43% - 80%	23% - 31%	30% - 44%	29% - 33%	15% - 8%	20% - 12%
Suicune	57% - 45%	40% - 50%	47% - 48%	57% - 45%	40% - 50%	47% - 48%	50% - 0%	20% - 0%	29% - 0%
Zapdos	0% - 75%	0% - 50%	0% - 60%	0% - 75%	0% - 50%	0% - 60%	0% - 12%	0% - 8%	0% - 10%
Overall Accuracy	All Features			Just Pixels			Just Extracted Data		
	No PCA = 30% With PCA (10) = 45%			No PCA = 30% With PCA (10) = 45%			No PCA = 25% With PCA (10) = 16%		

Table 7. KNN's classification report. Without PCA (blue colour) and with PCA (red colour)

For KNN we have set its default parameters thus using only 5 neighbours and uniform weights. In comparison to SVM, KNN's accuracy results for 'All Features' and 'Just Pixels' are exactly the same. However, these produced worse results than SVM's results except for the non-PCA version of this model. As I have mentioned in a previous section, KNN stores its training data which might be the cause for better results when the model runs without PCA. As we can see from Table 7, just like SVM, this model also performs better with PCA except for when the 'Extracted Features' data is used. However, unlike SVM, when PCA is not used for this data type, it performs worse. This was expected as KNN is portrayed as a 'lazy algorithm' since it would classify data based on the similarity of its neighbours. Our theory is that some Pokémon were classified poorly as some of them have similarities in terms of their shape and colour.

A discrepancy that we have identified, is that the model doesn't seem to classify the *Zapdos* class at all. Moreover, when PCA is applied, and the 'Just Extracted' data is used, *Suicune*, *Zapdos* and *Latios* are not classified at all. As I mentioned previously, this model classifies data based on the similarity of its neighbours. These 3 Pokémon share some features with each other such as their shape and colour which could cause these poor results when the model tries to classify them, please refer to Figure 11. in 'Figures' section.

Random Forest Classifier

No PCA- PCA	*All Features PCA n comp = 14*			*Just Pixels PCA n comp = 15*			*Extracted Features PCA n comp = 10*		
Class	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Articuno	29% - 50%	15% - 38%	20% - 43%	12% - 45%	8% - 38%	10% - 42%	10% - 15%	8% - 15%	9% - 15%
Entei	67% - 83%	60% - 50%	63% - 62%	56% - 83%	90% - 50%	69% - 62%	57% - 38%	40% - 50%	47% - 43%
Groudon	60% - 78%	50% - 58%	55% - 67%	75% - 64%	50% - 58%	60% - 61%	40% - 12%	17% - 8%	24% - 10%
Kyogre	31% - 56%	50% - 50%	55% - 53%	30% - 58%	30% - 70%	30% - 64%	24% - 33%	40% - 20%	30% - 25%
Latias	80% - 67%	29% - 43%	42% - 52%	75% - 62%	43% - 36%	55% - 45%	20% - 12%	14% - 14%	17% - 13%
Latios	5% - 9%	25% - 25%	8% - 13%	4% - 7%	25% - 25%	7% - 11%	0% - 0%	0% - 0%	0% - 0%
Moltres	55% - 30%	75% - 38%	63% - 33%	44% - 42%	50% - 62%	47% - 50%	43% - 20%	38% - 12%	40% - 15%
Raikou	15% - 23%	40% - 60%	22% - 33%	33% - 62%	57% - 71%	42% - 67%	0% - 0%	0% - 0%	0% - 0%
Rayquaza	57% - 47%	31% - 54%	40% - 50%	50% - 56%	17% - 42%	25% - 48%	18% - 25%	15% - 17%	17% - 20%
Suicune	60% - 60%	30% - 60%	40% - 60%	67% - 44%	33% - 33%	44% - 38%	22% - 14%	20% - 8%	21% - 11%
Zapdos	75% - 78%	50% - 58%	60% - 67%	80% - 56%	50% - 62%	62% - 59%	33% - 0%	17% - 0%	22% - 0%
Overall Accuracy	All Features			Just Pixels			Just Extracted Data		
	No PCA = 41% With PCA (14) = 50%			No PCA =40% With PCA (15) = 49%			No PCA = 20% With PCA (10) = 15%		

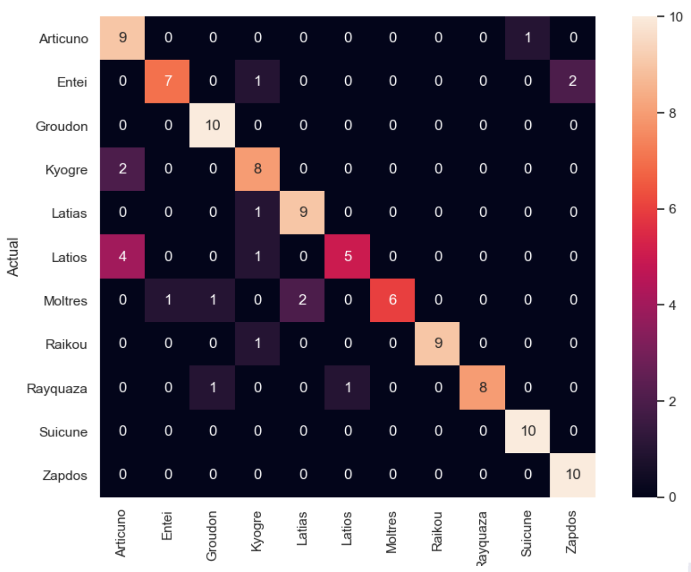
Table 8. Random Forest's classification report. Without PCA (blue colour) and with PCA (red colour)

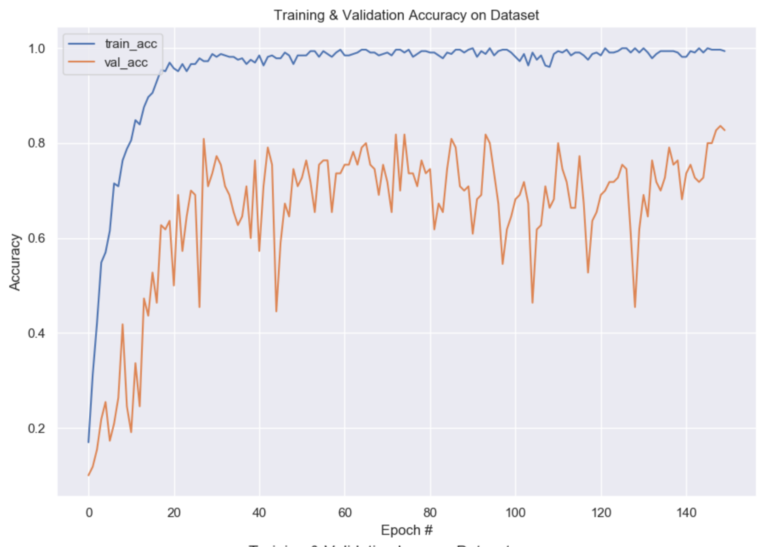
Our RF model uses 130 for its number of trees and different PCA n component for different data types, refer to Table 8. Across all the different data type with or without PCA, Random Forest classifier was not able to surpass the accuracy of any other classifier. Overall, when comparing the RF to the traditional classifiers it would always take the second place in terms of accuracy. From Table 8. we can see that on the data types all features and just pixels, the model performed better on a lower feature dimensionality. However, on the data type extracted feature, which has 12 features only, using a PCA affects the model negatively, our previous accuracy of PCA plots back this result. Also, note that the PCA in that instance had 10 component which is very close to the already 12 features the extracted features data type had.

The class type *Latios* is the most misclassified overall, when encountered the model would classify it correctly only 25% of the time and never on the just extracted features data type. In fact, we can see that all classes have an overall 57% or less across all 3 different calculations when using the just extracted features data type. *Raikou* is another highly misclassified class, checking its set of images, we see that many have elaborate backgrounds of black backgrounds, we suspect this could have affected its result.

Convolution Neural Network

Convolutional neural networks performed best across all classifiers used in this report, and without any extracted data. This again shows the power of the CNN and its randomly initialised kernels in extracting as much different representation as it can. Our CNN model consists of 4 convolutional layers with increasing number of filters and max pooling at each one. These are connected to a 4-layer neural network also with increasing number of neurons. The last convolutional layer has a dropout of 50% and all the neural layers but the last have a 25% dropout. We use ReLU and Softmax for our activation functions and Adam as our optimizer. Picture dimensions have been kept at 64x64 and the model ran for 150 epochs. Please refer to the python file titled “4 layer CNN.py” for more details on the hyperparameters used.





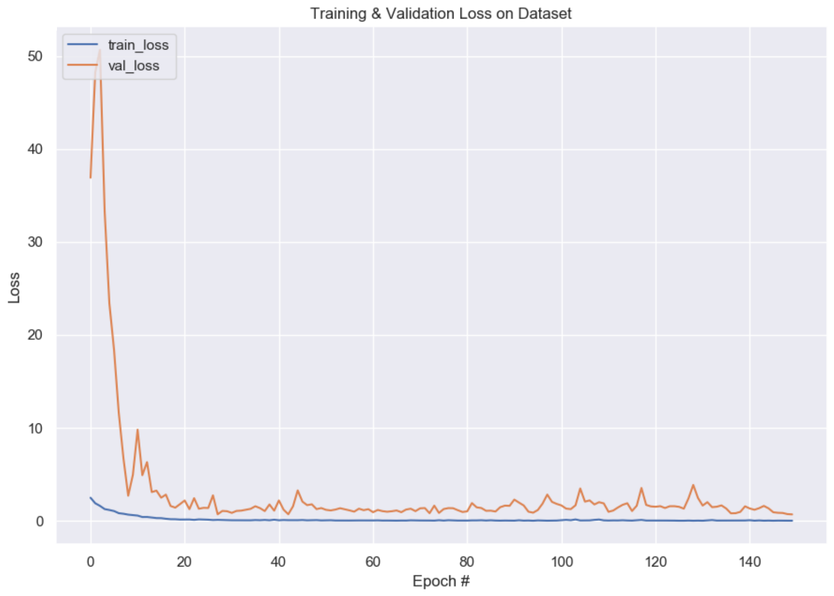


Figure 9. CNN’s Training & Validation loss over Epoch, Training and Validation Accuracy over Epoch, and Confusion matrix.

We can see on the loss plot, on Figure 9. that the model converges at around epoch 30, however we decided to go with bigger epoch in hope for the accuracy of the validation to stabilize a little. From the confusion matrix we can observe that only 3 classes were always correctly classified which are {*Groudon, Suicune, Zapdos*} this can also be seen in the classification report where the mentioned 3 classes have a recall of 100%, refer to Table 10. The class *Latios* had the lowest recall, meaning that the model was only able to predict it correctly 50% of the time when encountered which is not very good. However, this result correlates to what we have seen in the previous classifiers results. Moreover, the F1-score is relatively high across all classes, unlike in the other classifiers. The class {Suicune} comes at the top with the highest F1-score, meaning when encountered, it was correctly classified and when encountering different classes, it wasn't classified as it.

In conclusion, we can see that our hypothesis of the CNN outperforming the rest is true and that the SVM would indeed top the traditional classifiers given the optimal PCA configuration. We also saw the importance of pre-processing the data accordingly, in Figure 12. in the 'Figures' section. we see Pokémon's in different positions, art styles and backgrounds. Due to this randomness many aspects of shape and colour has been lost to the classifiers, leading to poor accuracy results. On the other hand, across all classifiers the class *Entei* has had the most stable scores, this could be due to it being the only Pokémon with predominately brown colours.

Problems or difficulties working with the dataset

Overall, all our classifiers ran quickly with our dataset and since the dataset was considerably small in terms of number of images and image dimensions, we saw no need to normalize the images beforehand. However, we did normalize the output from each layer in the CNN model, due to the nature of the ReLU function to explode. Recall, that the dataset was quite unbalanced and had many unneeded classes. As such, we had to manually re-search images of the different Pokémon's, modify them, then add them to our dataset. In terms of coding, there were no difficulty devising a method to operate on the dataset.

Class Name	*Precision*	*Recall*	*F1-score*
Articuno	60%	90%	72%
Entei	88%	70%	78%
Groudon	83%	100%	91%
Kyogre	67%	80%	73%
Latias	82%	90%	86%
Latios	83%	50%	62%
Moltres	100%	60%	75%
Raikou	100%	90%	95%
Rayquaza	100%	80%	89%
Suicune	91%	100%	95%
Zapdos	83%	100%	91%
Overall Accuracy	**83%**		

Table 10. CNN's classification report showing Precision, Recall, and F1-score for each class and the overall accuracy.

Figures

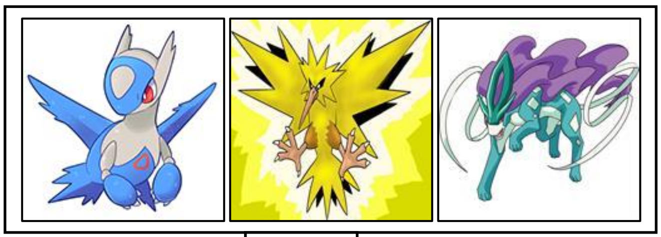


Fig 11.

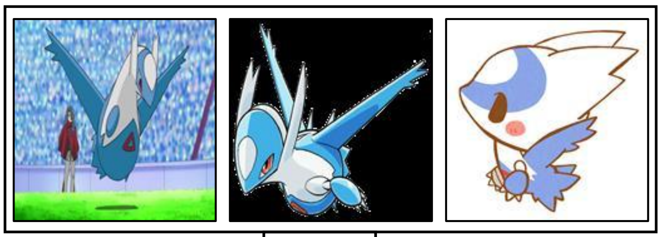


Fig 12.

Figures 11. above represent the *Latios*, *Zapdos* and *Suicune* classes respectively. Figure 12. represent images from the Latios' class in different positions, with and without intricate backgrounds, and in different art style.

References:

- Doctor Disney, (2014). Disney Cracking Down On Vendors Selling Unlicensed Merchandise. [Online] Available at: http://doctordisney.com/2014/08/19/disney-cracking-vendors-selling-unlicensed-merchandise/ [Accessed 11 Dec 2019]
- Polygon, (2019). Artists are provoking Nintendo and Disney to sue bots that steal artwork. [Online] Available at: https://www.polygon.com/2019/12/5/20997056/i-want-that-on-a-shirt-bots-copyright-infringement-twitter [Accessed 11 Dec 2019]
- Towards Data Science, (2018). SUPPORT VECTOR MACHINES (SVM) Available at: https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589 [Accessed 11 Dec 2019]
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. MIT Press. chapter 9, pp. 326
- Wikipedia, Feature engineering (2019) [online] Wikipedia.org Available at: https://en.wikipedia.org/wiki/Feature_engineering [Accessed 2 Dec 2019]
- Taloba, A., Eisa, D., Ismail, S. (2018) A Comparative Study on using Principle Component Analysis with different Text Classifiers. International Journal of Computer Applications, 180(31), pp.1–6.