Artificial Intelligence for Creative Technologies (UFCFSN-15-3) Image Classification

Thomas Pasley
19020362

Contents

1 Introduction

In this report, we will discuss the use of a convolutional neural network (CNN) to classify images of gemstones. This model was trained for the purpose of using it as a creative application tool, which would allow users to easily categorize and analyze gemstone images. We will discuss the process of training and evaluating the CNN, as well as its technical implementation and our results in viability of this model as an effective creative application.

2 Research

2.1 Classification

In machine learning, classification is the process of predicting the class or category that an input sample belongs to. This is done by training a classification model on a labeled dataset, where each sample in the data set is assigned to one of a set of predefined classes. The model then uses this training data to learn the patterns and relationships that exist between the input data and the corresponding classes.

Once the model has been trained, it can be used to make predictions on new, unseen data. This is done by providing the model with an input sample and having it output the class or category that it thinks the sample belongs to. The model makes these predictions based on the patterns and relationships that it learned during training.

There are many different types of classification algorithms that can be used in machine learning, including logistic regression, decision trees, and support vector machines. The choice of algorithm will depend on the specific characteristics of the data set and the prediction task at hand.

2.2 CNN's

Convolutional neural networks (CNNs) are a type of neural network that is commonly used for image classification. CNNs are trained to recognize patterns in images and can be used to classify images into different categories. This is done by using a series of filters to identify features in the images, such as edges and shapes, and then using these features to make predictions about the contents of the images.

CNNs are particularly useful for image classification because they are able to automatically learn the most important features in an image. Unlike traditional machine learning algorithms, which require manual feature extraction, CNNs learn to identify the important features in an image on their own. This

makes them well-suited for image classification tasks, where the features that are important for making predictions about an image may not be obvious.

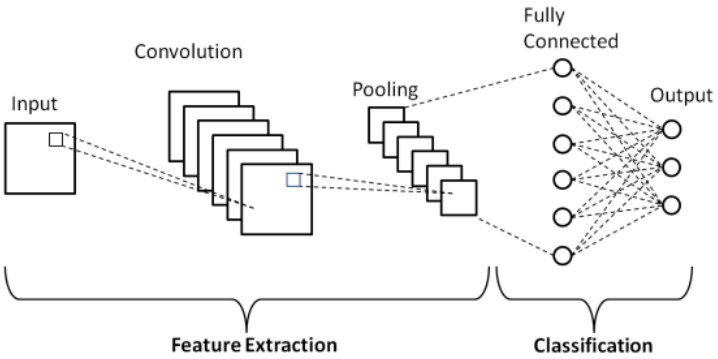

Figure 1: Diagram of how a CNN process an image input to output as prediction based on classes. Phung and Rhee (2019)

Another reason why CNNs are useful for image classification is that they can handle images of different sizes and orientations. Traditional machine learning algorithms require input data to be of a fixed size, which can be a problem when dealing with images, which can have varying sizes and aspect ratios. CNNs, on the other hand, are able to process images of different sizes by using a process called "pooling" to reduce the size of the input data without losing important information. This allows CNNs to be more flexible and robust when dealing with images. This was beneficial when training as due to the subject chosen, uniform data in format was not always available and we relied upon online sources to gather this as is discussed later in this report.

Overall, CNNs are an effective tool for image classification because they can automatically learn important features in an image and can handle images of varying sizes and orientations. This makes them well-suited for a wide range of image classification tasks. Therefore a CNN was to be used as the most suitable method for the intention of the creative application.

2.3 Learning Problems

One of the problems that can occur when training a convolutional neural network (CNN) is overfitting. Overfitting occurs when a model becomes too complex and starts to adapt too well to the training data and then performs poorly with new data the model hasn't seen before. This can lead to poor performance

on new, unseen data. This model was particularly prone to issues with overfitting due to the large size of its training data set and the diversity of appearances a class could have which were not always possible to accurately represent in its validation stage.

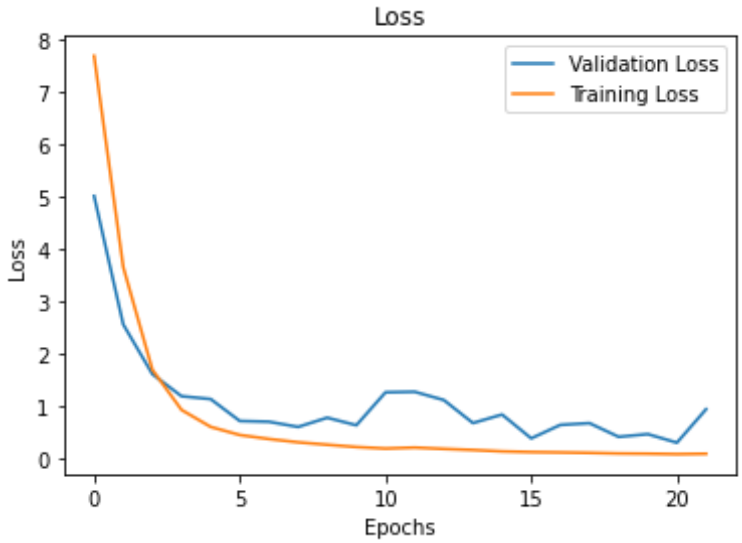

Figure 2: Example where validation loss has a notable offset from training loss otherwise known as over fitting.

2.4 Other Attempts

The popular data scientist and machine learning enthusiast site Kaggle hosts a few different models and notesbooks conducted on classifying gemstones, Chemkaeva (2020). Many others chose to use CNN's as well as transfer learning. Due to issues with certain gemstones being very similar in appearance, Amazonite with purple incandesance and Amethyst for example, others chose to focus on the augmentation of the data set to ensure their models would be able to correctly focus on features that would separate these. Other attempts also focused on using their model for classification purposes.

3 Methods

3.1 Model

Following the teaching from the module the classification model started as a simple CNN before employing method such as transfer learning for better accuracy. Eventually, the final model would use an 18 layer Resnet model and an Adam optimiser.

3.1.1 Transfer Learning

Transfer learning is a technique in which a model trained on one task is used as the starting point for a model on a second, related task. This is particularly useful when the second task has less data available for training than the first, as it allows the model to take advantage of the knowledge it has already learned from the first task.

In the case of convolutional neural networks (CNNs), transfer learning is often used to train models for image classification tasks. A common approach is to take a pre-trained CNN model that has already been trained on a large dataset, such as ImageNet, and use its learned features as the starting point for a new model that is trained on a smaller, related dataset. This allows the new model to take advantage of the features and patterns that the pre-trained model has learned from the larger dataset, which can improve the performance of the new model on the smaller dataset.

3.1.2 ResNet Model

ResNet is a convolutional neural network (CNN) architecture that was developed by researchers at Microsoft Research in 2015. It is known for its ability to train very deep neural networks, up to hundreds of layers, without suffering from the "vanishing gradient" problem that plagues many other deep learning models Shorten (2019). This is achieved through the use of "skip connections" that allow information to bypass one or more layers in the network, enabling faster and more effective learning. As a result, ResNet has become a popular choice for many classification tasks in computer vision, and has been used to achieve state-of-the-art performance on a number of image recognition benchmarks.

3.1.3 Model Augmentation

Data augmentation is a technique used to increase the amount of training data available to a machine learning model, particularly in the context of image recognition tasks. This is typically done by applying a variety of random transformations to the existing training data, We use flipping, cropping, rotating, and normalising the images, in order to create new, augmented versions of the original data. This was especially beneficial for our model. By using data augmentation methods, We were able to significantly the diversity of the training

data set, which can help to improve the generalization ability of our model and reduce over-fitting.

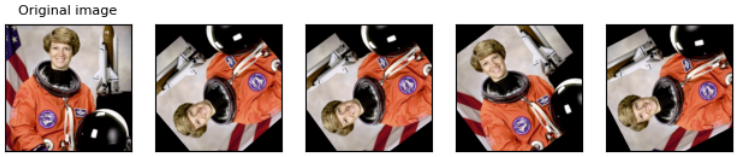

Figure 3: Example where a random rotation has been used to increase the diversity of training data. Torch (2017)

3.2 Data set

The first step in creating a suitable data set was automating collection of images. In certain cases, manual collection is suitable and preferred due to ensuring that the correct images are assigned to each class. However due to the amount of classes required for this model automation was employed to create a large enough data set to appropriately train this model.

The data set uses the recommended 70/15/15 split for its: Training, Validation, and Testing respectively.

3.2.1 Generation

A python script was used to pull images from the search engine Bing. The script would separate the images by search term which copied the classes. After this a verification process was started to sort the gathered images to remove any images that were not suitable or not of the subject at all. This resulted in slightly smaller data set which had minor differences in the amount of images per class but was now more suitable for training.

```
from bing_image_downloader import downloader

downloader.download("Gemstone Example Here", limit=100,
output_dir='dataset', adult_filter_off=True,
force_replace=False, timeout=60, verbose=True)
```

Figure 4: Bing Image Downloader Code Example, Developed by Singh (2020)

This initial data set was used for the first few weeks of development until it later proved that due to different image contexts that it was limited the accuracy of the model. The highest accuracy achievable was 52%. This raised a need for a more concise data set.

3.2.2 Editing

Our new data set was generated by only scraping images from websites specifically for hosting Gemstone Images. We would then use a python script that would remove the backgrounds of these images so the model would not be confused by common background occurrences such as metals or hands. This script parses an image to see where there is large change in pixels compared to the outer area of the image to find the subject of the image and remove the rest of the image. The images now would only have the class subject with a black background. An unforeseen advantage of this script was that it would also remove reflective artifacts of each image which helped to ensure the model would not see this as a distinctive feature since it is shared by most classes.

```
from PIL import Image
from rembg import remove
import os

directory = r'DATASET PATH GOES HERE'
terms = []
c=1
for filename in os.listdir(directory):
    #print(os.path.join(directory, filename))
    for item in os.listdir(os.path.join(directory, filename)):
        print(item)
        if item.endswith(".jpg")
        or item.endswith(".png")
        or item.endswith(".JPG"):
            print(item)
            im = Image.open(os.path.join(directory, filename, item))
            output = remove(im)
            name='img'+str(c)+'.png'
            output.save(os.path.join(directory, filename, name))
            os.remove(os.path.join(directory, filename, item))
            c+=1
            print(name)
            continue
        else:
            continue
```

Code used to removed backgrounds from data set Images

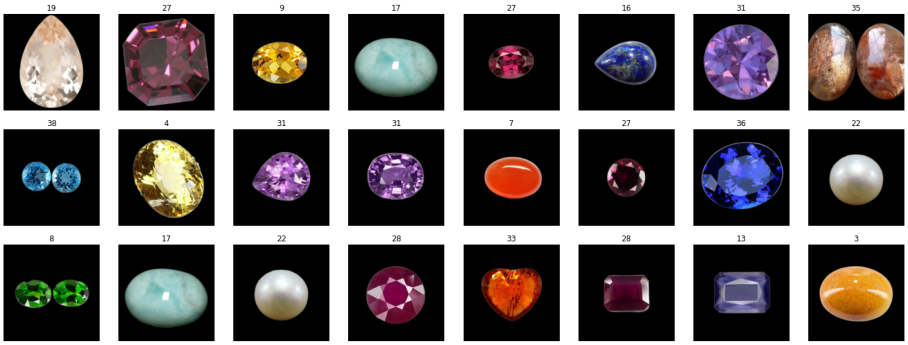

Figure 5: Example of Edited Data Set Images.

4 Technical Implementation

4.1 Pytorch

PyTorch is a popular deep learning framework for training and deploying neural networks. One of the key features of PyTorch is its ability to easily and efficiently build and train convolutional neural networks (CNNs), which are a type of neural network well-suited for image classification and other computer vision tasks. PyTorch provides a number of high-level and low-level APIs for building and training CNNs, making it a powerful tool for deep learning researchers and practitioners. Additionally, PyTorch's dynamic computational graph approach allows for efficient training of CNNs on large datasets, making it a valuable tool for anyone looking to build and train CNNs for a variety of applications.

4.2 Local Runtime

Initially Google Colab was used to create a runtime to train this model. However this quickly became inefficient as Google Colab had usage limits and limited processing power. The solution to this was to start training the model on a local runtime to utilize the extra CUDA cores without concern of approaching usage limits. The benefit of the extra CUDA cores also meant that the time taken on each epoch was reduced greatly. The steps used to create a local runtime were provided by Ahsan (2021).

By using Nvidia's CUDA Toolkit, Nvidia (2022), we were able to train the model on a local machine. Jupyter was used that Notebook Python would still be present. The process of enabling local training was ideal for amount of classes and images in each, as tweaking parameters for optimisation would not be feasible to test due to each run taking roughly 6 minutes per epoch.

4.3 Parameters

Best Model Parameters: Accuracy: 81%. Batch size: 48, Learning Rate: 0.0003, Number Of Epochs: 30.

5 Results and Findings

We were able to produce a model that was highly accurate at predicting Gemstones. The most beneficial methods were most importantly transfer learning and editing the data set to better reflect the subject of each class.

5.1 Accuracy

The final highest accuracy achievable was 81% when using the final edition of the data set. Noticeably the issues causing the decrease in accuracy was from similar classes where the model would lean in to one class more than the other causing a weakness in an entire class. Although we still result with high accuracy this would cause issues when applying the model as it would have a blind spot that would be often incorrect rather than occasional inconsistencies.

5.2 Application

In terms of using this model for a creative application as is the context of the module, The model would be more than sufficient for confident answers for classification. However, there would be constraints tied to this. The model has been trained on a very simple data set without complexity in backgrounds or obscuring of the subject. This would mean a user would need to provide their images in the same context to ensure the model provides the same accuracy as in testing.

Class weaknesses also provide issues in application. Users who would try to authenticate certain gemstones where the model has either not been trained or had difficulty assessing such as: Beryl Golden and Quartz Lemon. Overall, the accuracy and breadth of the model would make this suitable for use as an application but would benefit from deeper insights such as confidence on each prediction.

6 Conclusion and Future Improvement

In conclusion, we were able to create a accurate model that would suitable for a creative application. The models accuracy shows a great capability for classifying gemstones. Overall, despite the accuracy of the model short comings such as failure on an entire classes would cause issue in real world application as discussed.

In future, the best path of improvement would be to work on the models augmentation. Currently augmentation can have an affect or distinguishing features of data such as colour and reflections. Making alterations to how data augmented to preserve these features would likely not only increase accuracy but help the model differentiate between similar classes.

References

N. Ahsan. How i setup my windows machine for deep learning, Jul 2021. URL https://medium.com/analytics-vidhya/how-i-setup-my-windows-machine-for-deep-learning-82358a939c74.

D. Chemkaeva. Gemstones images, Mar 2020. URL https://www.kaggle.com/datasets/lsind18/gemstones-images.

C. Nvidia. Cuda toolkit - free tools and training, Dec 2022. URL https://developer.nvidia.com/cuda-toolkit.

V. H. Phung and E. J. Rhee. A high-accuracy model average ensemble of convolutional neural networks ..., Oct 2019. URL https://www.researchgate.net/publication/336805909_A_High-Accuracy_Model_Average_Ensemble_of_Convolutional_Neural_Networks_for_Classification_of_Cloud_Image_Patches_on_Small_Datasets.

C. Shorten. Introduction to resnets, May 2019. URL https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4.

G. P. Singh. Bing-image-downloader, May 2020. URL https://pypi.org/project/bing-image-downloader/.

C. Torch. Rotate, 2017. URL https://pytorch.org/vision/stable/generated/torchvision.transforms.functional.rotate.html.