Final Project for Machine Vision Course (Image Classification approaches Using ML and DL)

Ahmed Asaad (G1916043)

1. Introduction

Image classification is a primary task in the field of Computer Vision. A classifier is used to receive an image as an input and label this image with the corresponding class as an output. Before the classification stage, image features should be extracted to act as parameters to the classification function. Many algorithms can be used as classifiers. Machine learning and Deep Neural Networks algorithms are the most methods used in classification. Convolutional Neural Networks CNN are the state of art algorithms which demonstrated better performance with high accuracy results.

In this project, I worked on image classification task using virous methods like Bag of Words and SVM classifier, Feedforward NN and CNN. In addition, I used a transfer learning method where a very well pre-trained model is used to do the classification. Here, the same parameters of the model will be used in the new task while only the last layer is replaced with the new task number of classes. A fine-tuned method is also implemented by learning the well trained neurons more to get better results.

For each method, several experiments done to algorithm by changing parameters and the using different functions those used in the algorithms. By doing so the performances of the algorithm are compared to each other to see how these parameters and functions affects the final result.

2. Dataset:

A five classes of jewelry dataset is used in this project. These classes are (ring – necklace – bracelet – earring – nose ring). This dataset is quit challenging due to the close similarities between them. In each class I collected 105 images for the training dataset and 45 for the testing dataset. By using separated code, I increased the dataset by implementing data augmentation to each image. The number of images per class after the data augmentation became 630 images for the training dataset and 270 images for the testing dataset. Some of the image transformation are done to the images. Rotation by 90 and 45 degrees, translating image to the right by the quarter of the width and the quarter of the length to the bottom, flipping the image upside down and finally transforming the image by changing the position using two arrays.

3. Classification Algorithms:

3.1. Bag of word:

Bag of Word or Bag of Visual word is used in image classification. The idea came from the document classification where the keywords in a document are counted and a frequency histogram for each keyword is made. In the case of images, the feature of the image are the words. The general idea of bag of visual words (BOVW) is to represent an image as a set of features. Features consists of keypoints and descriptors. Keypoints are the “stand out” points in an image and descriptor is the description of the keypoint. We use the keypoints and descriptors to construct vocabularies and represent each image as a frequency histogram of features that are in the image. From the frequency histogram, later, we can find another similar images or predict the category of the image.

To detect the features and extract the descriptors, we can use feature extractor algorithms like (SIFT, SURF and local HOG). Then the descriptors of the features are clustered by clustering algorithm such as (K-Means, GMM). The center of each cluster will be used as the visual dictionary’s vocabularies.

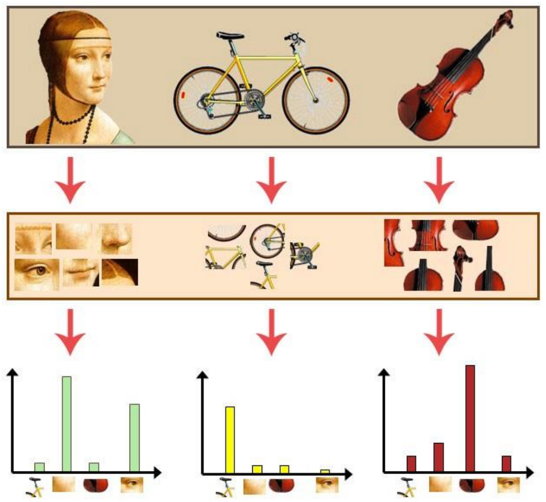


Figure (1) - BOVW

Finally, for each image, we make frequency histogram from the vocabularies and the frequency of the vocabularies in the image. Those histograms are bag of visual words (BOVW) - Figure (1).

The K-Mean clustring algorithm starts by choosing K points as the number of clusters wanted to represent the dictionary's vocabularies. Each K is the centroid of each cluster. Then all descriptors will be assigned to the closet centroids. Next step will iteratively recompute the centroid of each cluster and stop when centroid does not stop.

To encode novel images, we find the frequency (histogram) of local descriptors associated to each visual vocabulary (centroid of k-means). Each image then is represented by a k-dimensional vector. Each value in the vector is the number of descriptors assigned to the cluster.

The classifier SVM which is one of the most robust prediction methods that analyze data for classification and regression analysis. It receives the training images and apply a linear classification to the different labeled features. The idea is to maximize the width of the gap between the categories. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

3.2. Fully connected Neural Networks (FNNs)

FNN is the simplest form of artificial neural network. FNN uses the deep learning to perform classification.

In this network, the information moves in only one direction—forward—from the input nodes (neurons), through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network. We can see the structure of FNN in figure (2). IN our case (image classification), the input is an image which is flattened to be fed through input layer and the output represents the required classes. FNNs primarily used for supervised learning. That is, feedforward neural networks compute a function *f* on fixed size input *x* such that *f*(*x*)≈*y* for training pairs (*x*,*y*).

The basic unit in NN is the neuron. An artificial neuron takes on an input vector and outputs a scalar value. The neuron is parameterized by a set of weights. Each weight is used as a multiplier for a scalar input. The output of the neuron is the result of applying a nonlinear activation function on the sum of the weighted inputs. Thus, a neuron with weights w, inputs x, output y, and non-linear activation function f is represented as:

$$y = \emptyset\left((\sum_{i=0}^{n} w_i \cdot x_{i)+bias} \right)$$

Where n is the number of layer inputs, x is the input and w is the associated weight. Bias is the neuron's controlling value which decide if this neuron is going to give an output or not.

Our purpose is to learn the weights and the biases in order to

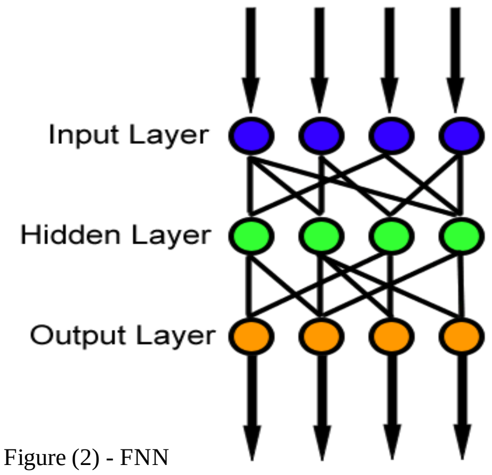


Figure (2) - FNN

have a correct output or mathematically to reduce the output error. And since the small change in weights and biases may lead to a big change to the output, non-linear activation functions f (x) are applied to the neuron term. An additional requirement of the learning algorithms over f (x) is differentiability. Some of the nonlinear activation functions are sigmoid, Relu and tanh.

An artificial neural network (ANN) consists of a set of connected neurons. Typically, neurons are grouped in layers. Each layer takes a set of inputs for computing a set of outputs. The input-output relation is determined by the weights in the layer.

DNNs are usually trained using supervised learning. The objective of the training procedure is to find the network parameters that minimize a loss function. The approach normally used is gradient based approach which iteratively optimize the output. The output of the network is back propagated through the network layers to update the neuron's parameters in order to have the minima loss based on gradient based method.

For image classification task, input is an image and output is the label or class of the image. The process of the training is the same as mentioned before where the ultimate training ensures the class of the image is correct. We can play around with some parameters in the algorithm to enhance the training process such as changing the learning rate, choosing different non-linear activation function or maybe using dropout technique.

Dropout technique consists of using only a random subset of the neurons in each layer (along with its connections) during training episodes in order to reduce overfitting. This procedure is able to improve the generalization error of the networks.

3.3. Convolutional Neural Networks CNNs

CNNs are a class of Deep Neural Networks which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

They are different from the FNN by not taking all layers to be fully connected. Instead, some of the layers in starting from the input layer employ a mathematical operation called convolution. The convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a ReLU layer, and is subsequently followed by additional convolutions such as pooling layers.Fully connected layers are connected sequentially with the last convolution layer. CNN referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution.

The CNN classification algorithm is similar to FNN in taking an image as an input, feedforward, computing the loss and then backpropagating the loss to update the parameters and finally reaching the minima loss. The difference is in the architecture of the network. Instead of having the input image as number of pixels, ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction.

The architecture of the ConvNet is shown in figure (3). First is the stage of feature learning where the convolutional layers exisit and second stage is for the fully connected layers to perform classification.

In details we can take a look at the components in the CNN:

Convolutional layer:

The input of the CNN is an image with shape (image height) x (image width) x (input channels). Convolutional layers convolve the input and pass its result to the next layer. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter,K. We select K as a nxnx1 matrix. The filter is applied to the image and progressively convolve the input image and move from up left to down right in steps of the stride length until it finish the process. The output is a feature map with the size of nXnXnumber of filters. Where

$$n = [(W-K+2P)/S]+1 \quad .$$

W is the input width, (32)

K is the Kernel size (3)

P is the padding (0)

S is the stride (default = 1)

The convolution operation brings a solution to this problem as it reduces the number of free parameters (weights), allowing the network to be deeper with fewer parameters. For instance, regardless of image size, kernel of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters.

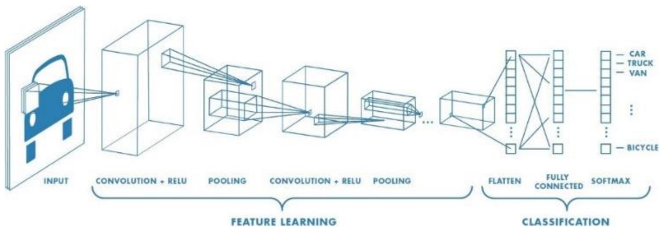


Figure (3) – CNN architecture

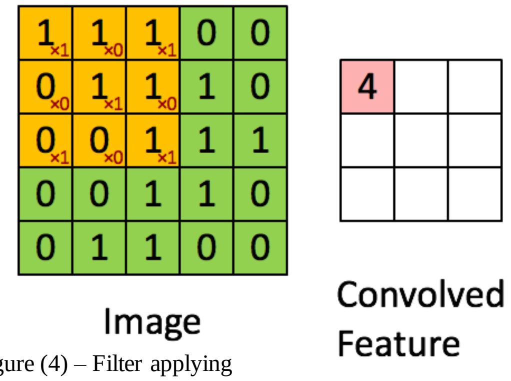


Figure (4) – Filter applying

The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image-figure(4). ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network, which has the wholesome understanding of images in the dataset, similar to how we would.

Pooling Layer

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved

Feature. This is to decrease the computational power required to process the data through dimensionality reduction.

There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling - figure (5) returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

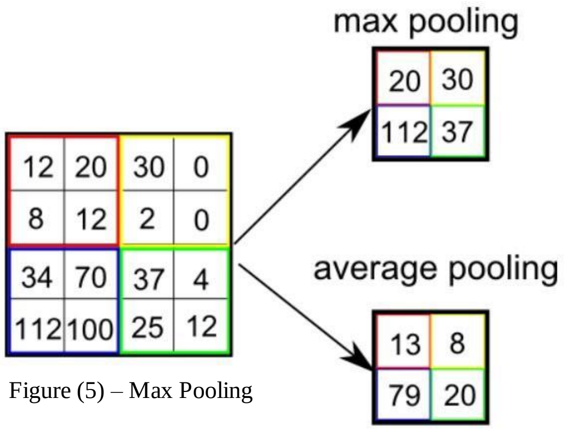


Figure (5) – Max Pooling

After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

3.4. Classification — Fully Connected Layer (FC Layer)

The fully connected layer receive the output from last convolution layer as an input. The input image is flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the **Softmax Classification** technique.

3.5. Transfer Learning:

In CNN, we can apply the knowledge learned in previous task to novel ones. In Transfer Learning we take advantage of the very well-trained models to perform a task (classification in our case). The benefit of this is to get faster performance and accurate result. In addition, less training data is needed to do the training. So instead of learning a CNN from scratch we just use the pre-trained model and feedforward our dataset. The only adjusting is that the last layers (the ones that make the final classification) are replaced and whose first layers are re-used. The same features can be used for solving the two different problems. And that the classifiers (output layer) need to be trained again only while freezing the learning of parameters of the reused layers.

Fine-tuning

In fine-tuning, the same concept of Transfer learning is applied except we need to train the pre-trained parameters more as well as the new used layer(s). So the new layers must always be trained by using the second set of objects. Also, the reused layers can be trained further with a smaller learning

rate depending on data availability. In

general, we can set learning rates to be different for each layer to find a tradeoff between freezing and fine-tuning.

4. Results:

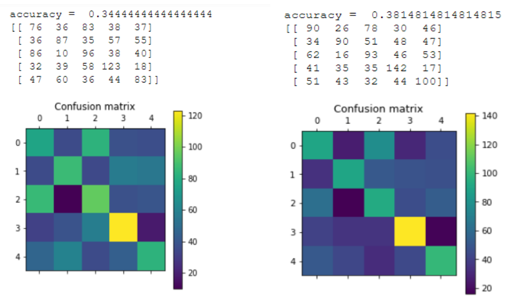
1. Bag of Word:

The first algorithm used in this project is Bag of word. It is an algorithm which detects and extracts features of the image and then uses an ML classifier to classify the image. In the code I used two different feature extractor (SIFT and BRISK) and in the encoding stage when clustering the features and make a vocabulary dictionary I used K-Means unsupervised algorithm with different values to see how accuracy is affected. I also used the SVM and KNN classifiers.

1.1. Comparative analysis of different classifiers:

In this section, I used two ML classifiers (SVM and KNN) and two feature extractors (SIFT and BRISK). The Number of clusters used in the code is 50 and 300. I could not go for more K because that made the code running without an end! This may be because the speed of my device. The results of the accuracy are reported in the table. I have included a sample for the confusion matrix for the BRISK-SVM in the case of 50 and 300 number of clusters(K). The result from KNN classifier showed non-reliability. In the case when number of classes increased, it gave less accuracy likewise the SVM.

K / Method		50	300
BRISK	SVM	0.34	0.38
	KNN	0.33	0.32
SIFT	SVM	0.41	0.39
	KNN	0.39	0.29



```
accuracy =  0.34444444444444444
[[ 76  36  83  38  37]
 [ 36  87  35  57  55]
 [ 86  10  96  38  40]
 [ 32  39  58 123  18]
 [ 47  60  36  44  83]]
```

```
accuracy =  0.3814814814814815
[[ 90  26  78  30  46]
 [ 34  90  51  48  47]
 [ 62  16  93  46  53]
 [ 41  35  35 142  17]
 [ 51  43  32  44 100]]
```

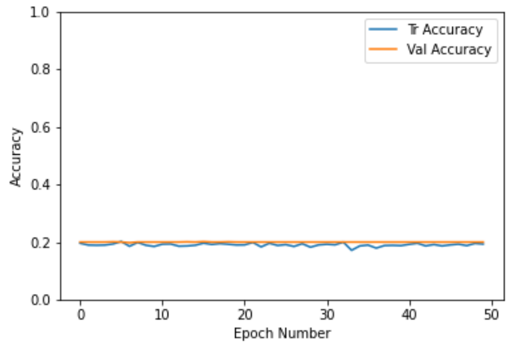
Confusion Matrices for K=50 and K=300 using Brisk feature extraction

2. Classification accuracy using Fully Connected Neural Networks

2.1. Comparative analysis of different architecture

Here I minimized the number of neurons in each layers.

Architectures	Training Acc.	Testing Acc.
Default	20 %	20%
reducing number of neurons	19%	20%



2.2. Comparative analysis between using Sigmoid, tanh, and ReLU as activation functions

Activation function	Training Acc.	Testing Acc.
Relu	20 %	20%
tanh	20%	20%
sigmoid	20%	20%

2.3. Effects of Dropout:

For my dataset there was no affect of applying the Dropout or not. Both gave me the same accuracy 20%.

2.4. Stochastic Gradient Descent VS ADAM:

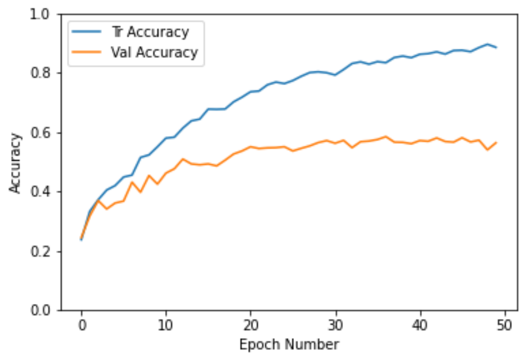
This also did not affect the accuracy of the training and testing dataset.

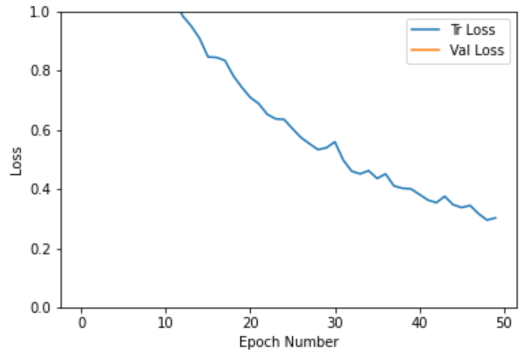
3. Classification accuracy using Convolutional Neural Networks.

3.1. Comparative analysis of different architectures:

With default considered architecture cosisting of 3 convolutional layers with pooling after each one and 3 fully connected layers(input, output and 1 hidden), I tried to change the architecture and see what will happen. The first change was to reduce convolutional layers to only 2 and keeping the fully connected layers the same.

architectures	Training		Testing	
	Acc.	Loss	Acc.	Loss
Default	89%	0.2	53%	2.1
The modified architecture	88.6	0.3	57%	1.8%





3.2. Comparative analysis between using Sigmoid, tanh, and ReLU as activation functions

Activation function	Training		Testing	
	Acc.	Loss	Acc.	Loss
Relu	89%	0.2	53%	2.1
Sigmoid	41.8%	1.3	43.1	1.3
tanh	91%	0.25	56%	2

3.3. Effects of Dropout:

Existence of Dropout	Training		Testing	
	Acc.	Loss	Acc.	Loss
With Dropout	89%	0.2	53%	2.1
With (0.5) Dropout	89%	0.29	55%	2.2

3.4. Stochastic Gradient Descent VS ADAM:

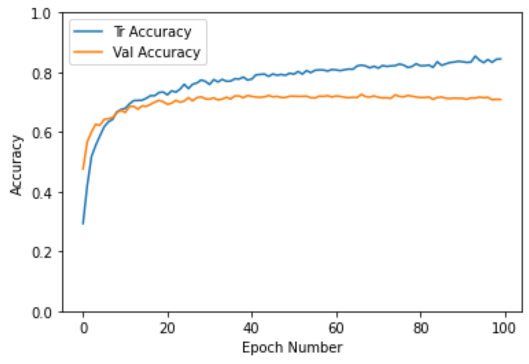
Here I applied different optimizers with the default architecture with the present od the Dropout.

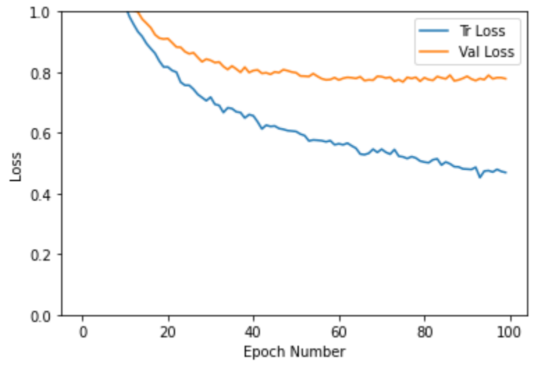
Optimizer	Training		Testing	
	Acc.	Loss	Acc.	Loss
ADAM	83%	0.4	57%	1.5
SGD	89%	0.2	53%	2.1

4. Classification accuracy using Transfer Learning and fine-tune methods:

Here I am using the pre-trained model (ResNet152) to apply the transfer learning. We keep the learning rate at zero or we freez the model parameter to prevent learning the model more. We just replace the classification layer which contains of 1000 neurons (The classes of ImageNet) with our number of classes (five). The number of epochs are 100.
For the fine-tune, we do exactly the same thing except that we keep the model parameters learning by setting `p.requires_grad = True`
The accuracy result was as follows:

Method	Training		Testing	
	Acc.	Loss	Acc.	Loss
Transfer Learning	84%	0.4	71%	0.7
Fine-tune	99.9	0.002	81%	0.9





5. Conclusion

The image classification is a fundamental task in Machine vision. Many algorithms can be used to perform such task. In this report I introduced some of these algorithms with short brief for each and reporting the results from the implementation of my own dataset (jewelry dataset) that I have collected. The algorithms used were from both ML and DL. The results showed that For ML algorithms, the accuracy was low comparative to the DL algorithms except for the FNN which also gave a very low accuracy for the testing dataset. The very low accuracy with FNN could be because of the challenging dataset I had In addition to the nature of the FNN which can not preserve the 2D spatial structure of the image. The best performance I got was from the fine-tune CNN which resulted with accuracy of 99.9 for training dataset and 81% accuracy for the testing dataset.