Emotion and Hand Recognition with CNN

Sangeeta Kumawat   
Computer Science

Kent State UniversityKent, Ohio USA  
skumawat@kent.edu

Jenifer Werthman  
Computer Science  
Kent State UniversityKent, Ohio USA  
jwerthma@kent.edu

*Abstract*

This paper illustrates the study of image classification with convolutional neural network. CNN is part of deep learning that is specifically designed to process and analyze image data. CNNs use convolutional layers to detect patterns and features in the input image, and pooling layers to reduce the dimensionality of the output. This allows CNNs to achieve high accuracy in image classification, object detection, and other computer vision tasks. It describes how to classify emotions and hands class based on the given datasets. Two different dataset, Kaggle emotions recognition and custom hand touch is used to test the CNN model for the accuracy and loss. The emotion recognition dataset has 7 classes (Anger, Happy, Disgust, Fear, Sad, Surprise and Neutral) and hand touch recognition dataset has 3 classes (Touch, No hands, No Touch without hands). To detect the classes, we have built the CNN with four convolutional layers and two fully connected layers.

Keywords—CNN, emotion, hand, touching, object detection, classification, deep learning

# Introduction

Image classification is the process of assigning one or more labels or categories to an input image based on its content. It is done using machine learning techniques, such as deep learning, where a model is trained on large dataset of labeled images to learn to recognize patterns and features in images that corresponds to specific categories or labels. Once trained, the model can be used to classify new, unlabeled images with high accuracy. Image classification has a wide range of applications, including object recognition, face recognition, medical imaging, and surveillance. One key features of deep learning in image classification convolutional architectures.

# Convolutional Neural Networks (CNN)

## Summary

Convolutional Neural Networks (CNN) are an algorithm based on the human brain inspired by the organization of the Visual Cortex developed in the 1980’s. An image is broken down into a matrix of information and pixel values. As you can imagine, this process will produce a lot of data; therefore, the larger the images and the number of images will become computationally intensive. CNN’s have several standard layers, though the methods to process the layers may vary: Input, Convolution Layer, Max Pooling, Fully Connected layer, Output. This algorithm requires a large amount of data to train which means a large computing resources.

## Input

The input layer takes in the train and test images and transforms them into normalized matrixes of information pertaining to each image.

## Convolution Layer

These are typically within what we would call the hidden layers. There could be 1 or more convolution layers used. These layers are the main layers of the network which contain the kernels, parameters, and values used throughout the training. This layer creates an activation map by multiplying the input image by the kernel for every element in the input image matrix. An Activation Map (otherwise known as a feature map) shows the most important areas of an image for a particular prediction.

## Pooling

Three types of pooling is Max Pooling, Min Polling, and Average Pooling. Max Pooling is a pooling operation that calculates the maximum value for each section of the feature map. Min Polling find the minimum value in each section of the activation map. Average Pooling finds the average value for each section of the activation map.

This layer helps to locate share and smooth features for edge and point detection.

## Fully Connected

This is where each neuron transforms the input vector through a weight’s matrix. All possible neurons are connection layer to layer. The result is that every input will influence the output. This allows the network to learn non-linear combinations for each feature.

## Output

The output of the CNN is a matrix/array the same batch size as the input but other information will/may change.

# Image Classification Solution

For the image classification, is a supervised model which allows us to train a model of images with labels and then predict what the classification of new images will be. For this project our goal was to build a generic model and then determine what improved or worsened the accuracy.

## Initial Analysis

Image Classification is time consuming and resource consuming. The first thing we did was start simple with a reduced dataset and few layers and epochs.

## Epochs

Epochs are defined as a period of time and in Machine Learning it relates to how many times we run our data through the training model. For our analysis we started with 1 Epoch and 5 Epochs to save time and then expanded up to 50 epochs. Our conclusion indicated that the more Epochs we ran our data through, the greater the accuracy.

## Optimizer

Optimizer is used to improve accuracy and reduce loss by choosing appropriate weights for the model. In our solution we choose to use the Adam optimizer.

## Batch Size

Batch size is the number of training elements to be used in one iteration. Batch size should be a power of 2, therefore we used varied numbers based on our data sizes. The smaller the batch the longer it takes to run, as there are more iterations to complete within one Epoch.

## Challenges

We encountered many issues initially where some examples would work on one platform but not on another. Once we did get things working we would get misc. random errors at various points.

Once we finally got it running, the next challenge was to figure out how to read images from a file folder rather than an API. After some trials and errors and choose to use the ImageDataGenerator to complete this task even though it is deprecating soon, it was easy to setup and we have both used it in the past.

For the hands-touch dataset, the original construction of the images presented a challenge for us as the data needed to be combined and organized into the classes. This took several hours. Ultimately, a script was created to help us organize them but with the images containing the same name we had to redo the effort in order to obtain the missing images.

Our final challenge was time. Once we begin seeing run times of 3.5 hours+ for the datasets that we were provided for the emotion dataset it became tedious to perform trials and errors on the dataset.

## Final Results

Our final results were fairly successful with a 99% accuracy rate for both Emotion Images and Touch Images; however, we did notice that our validation accuracy hovered between 40%-60%.

# Emotion Classification

## Summary

The emotion classification is comparing facial expressions in relation to emotion.

## Data

This dataset was fairly robust with a total of 28,821 images used in training and 7,066 images used in the validation.

|  |  |
| --- | --- |
| 3993 | Angry |
| 436 | Disgust |
| 4103 | Fear |
| 7164 | Happy |
| 4982 | Neutral |
| 4938 | Sad |
| 3205 | Surprised |

Figure 4-1: Data Categories

## Figures and Results

Text

Description automatically generated with low confidence

Figure 4-2: Training Results

Chart, line chart

Description automatically generated

Figure 4-3: Using Adam Optimizer Training and Validation Loss

Graphical user interface, application, Teams

Description automatically generated

Figure 4-4: Confusion Matrix

## Conclusion

For our solution we performed 4 Convolution Layers using the ReLU activation function, Max Pooling, and a drop out of .25. Additionally, we added 2 Fully Connected Layers with a drop out rate of .5 and Adam Optimization through 48 Epochs. This rendered the results of 99.56% accuracy with a loss of .26. Validation had an accuracy of 64.29% with a loss of 1.32.

# Hands-Touch Classification

## Summary

The hands touch classification is comparing images with hands touching and not touching with images without hands.

## Data

The hands touch dataset had a total of 13,153 images used in training and 2,052 used in validation.

|  |  |
| --- | --- |
| 4311 | No Hands |
| 4475 | No Touch |
| 4367 | Touch |

Figure 5-1: Data Categories

## Figures and Results

Table

Description automatically generated

Figure 5-2: Training Results

Chart, line chart, histogram

Description automatically generated

Figure 5-3: Using Adam Optimizer Training and Validation Loss

Chart

Description automatically generated

Figure 5-4: Confusion Matrix

## Conclusion

For our solution we performed 4 Convolution Layers using the ReLU activation function, Max Pooling, and a drop out of .25. Additionally, we added 2 Fully Connected Layers with a drop out rate of .5 and Adam Optimization through 20 Epochs. This rendered the results of 98.35% accuracy with a loss of .04683. Validation had a accuracy of 100% with a loss of .0033.

We did not notice a difference using this data set between having 2 convolution layers and 4. Both yeilded very similar results.

##### Team Roles

Our team consisted of two team members, Sangeeta Kumawat and Jenifer Werthman. For this project we worked together in person for initial planning and experimentation phases. Then due to time and resource constraints on running the models we split up and worked remotely.

Jenifer conducted the training over the emotion classifications, and started the report with findings. Jenifer also ran the final run of the hands-touch classification and shared results. Finally, Jenifer added elements to the report for CNN, Image Classification, Emotion, and Touch.

Sangeeta ran prelim results and experimentation over the emotion dataset and prepared the data for the hands-touch classification. Sangeeta also made necessary changes to our solution to accommodate the hands-touch data. Finally, Sangeeta completed the Introduction to the report and added additional details and information to remaining sections.

##### References

For this analysis we analyzed code provided by our professor Dr. JoonYoon Kim as well as several Kaggle references. Ultimately, we combined several ideas to form our finished resources for this project.

1. A.Krizhevsky, D.Sutskever, G.Hinton: ImageNet Classification with Deep Convolutional Neural Networks.
2. B. Reddy, P. Yashwanthsaai, A. Raja, A. Jagarlamudi, N .Leeladhar, T. Kumar: Emotion Recognition Based on Convolutional Neural Network (CNN)