FLOWER CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK

Author
Ayça KULA

ABSTRACT

Image classification is one of the most challenging and widely used task in the field of computer vision. Image classification is classifying images from many predefined categories. Many deep neural network algorithms have been used for image classification such as Convolutional Neural Network,Deep Belief Network and also machine learning algorithms as SVM, Random Forest etc.The objective the work presented in this paper, is to apply the concept the Deep Learning algorithm namely, Convolutional neural networks (CNN) in image classification and to discuss the results obtained from this algorithm. In order to implement this algorithm, Kaggle's flower dataset having 4242 images of 5 categories have been used. Classifying flowers has an extra challenge over other categories due to of the large similarity between classes. The experimental result analysis shows that the algorithm (CNN) gives fairly good classification accuracy for the tested dataset.

1 INTRODUCTION

Convolutional neural networks (CNNs) with more hidden layers have more complex network structure and more powerful feature learning and feature expression abilities than traditional machine learning methods(Alsaffar et al., 2017).Value of using GPUs for machine learning, modern GPU computing and parallel computing methods have increased the ability to train CNN models (Steinkraus et al., 2005). Also, the huge data development made CNNs even more popular and lead to their rise in research and industry. Image classification can be considered as the fundamental area for computer vision problems. Through the implementation and application of the neural network algorithm, we wish to show the neural network design for the image classification issue using Kaggle's flower dataset (Mamaev, 2018).

2 CONVOLUTIONAL NEURAL NETWORK

The aim of a neural network is to have a model that performs well both on the data that we used to train it (training dataset) and the new data on which the model will be used to make predictions (test dataset) (Goodfellow et al., 2017). Our paper uses a similar model seen in figure 1. However our model have more layers and our model can be seen from table 1.

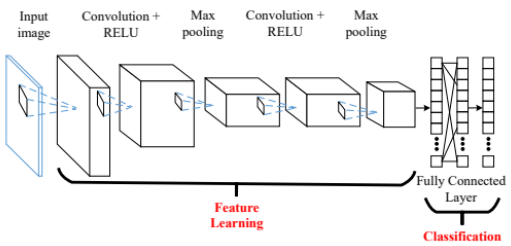

Figure 1: Convolutional Neural Networks (CNN) (Kamencay et al., 2017)

LAYER	TYPE	PROPERTIES	OUTPUT SHAPE	PARAMETERS
Layer 1	INPUT LAYER	224 x 224 x 3	(None, 224, 224, 3)	0
Layer 2	1 Convolutional	5X5 kernel dimension	(None, 224, 224, 32)	2432
Layer 3	Pooling	2x2 kernel dimension	(None, 112, 112, 32)	0
Layer 4	2 Convolutional	3x3 kernel dimension	(None, 112, 112, 64)	18496
Layer 5	Pooling	2x2 kernel dimension	(None, 56, 56, 64)	0
Layer 6	3 Convolutional	3x3 kernel dimension	(None, 56, 56, 96)	55392
Layer 7	Pooling	2x2 kernel dimension	(None, 28, 28, 96)	0
Layer 8	4 Convolutional	3x3 kernel dimension	(None, 28, 28, 96)	83040
Layer 9	Pooling	2x2 kernel dimension	(None, 14, 14, 96)	0
Layer 10	flatten (Flatten)	Multiple number of neurons	(None, 18816)	0
Layer 11	1 Fully connected - Dense		(None, 512)	9634304
Layer 12	2 Fully connected - Dense	5 classes	(None, 5)	2565

Table 1: The proposed Convolutional Neural Network Architecture

2.1 TYPES OF LAYERS

We have used three main types of layers to build CNN architectures: Convolution (CONV) Layer, Pooling Layer, and Fully-Connected Layer.When classifying images, the input to the first layer is the input image , while the output of the final layer is a set of likelihoods of the different categories.

2.1.1 INPUT LAYER

In CNN input layer should contain image data. Image data is represented by three dimensional matrix. It holds the raw pixel values of the image, in our case an image of width 224, height 224.

2.1.2 CONVOLUTION LAYER

The convolution layer uses filters that perform convolution operation as it is scanning the input with respect to its dimensions (che).The output is named as feature map or activation map. In this paper four convolution layers are used to design the whole process. Convolution layers also includes ReLU activation to make all negative value to zero (dshahid380, 2019).

2.1.3 POOLING LAYER

The pooling layer (POOL) is a downsampling operation (see figure 2), typically applied after a convolution layer, which does some spatial invariance.It makes the representations smaller and more managable.

Max and average pooling are special kinds of pooling where the maximum and average value is taken, respectively.In this paper after applying each of the convolution layers, max-pooling is used where each pooling operation selects the maximum value of the current view.The size of the pooling operation or filter is less than the size of the feature map and it is almost always 2×2 pixels applied with a stride of 2 pixels (Brownlee, 2019).Pooling layer (see Fig. 3) downsamples the volume spatially, independently in each depth slice of the input volume.The input volume of size [224 × 224 × 64] is pooled with filter size 2, stride 2 into output volume of size [112 × 112 × 64] (see Fig. 3). Notice that the volume depth is preserved.

2.1.4 FULLY CONNECTED LAYER

The fully connected layer (FC) operates on a flattened input where each input is connected to all neurons (che). If present, FC layers are usually found towards the end of CNN architectures and can be used to optimize objectives such as class scores. Computes the class scores where each of the numbers corresponds to a class score. As with ordinary Neural Networks and as the name implies, each neuron in this layer is connected to all the neurons in the previous volume (Kamencay et al., 2017).

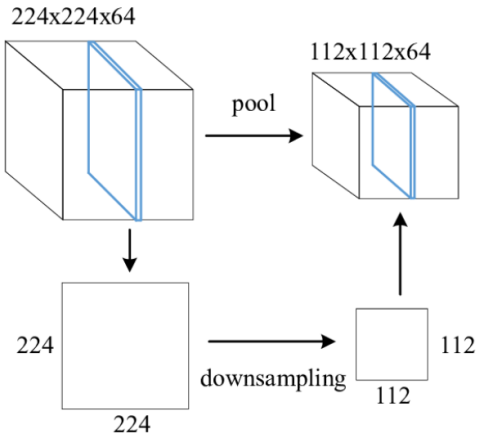

Figure 2: A pooling layer reduces the dimensionality of each feature map separately (Kamencay et al., 2017)

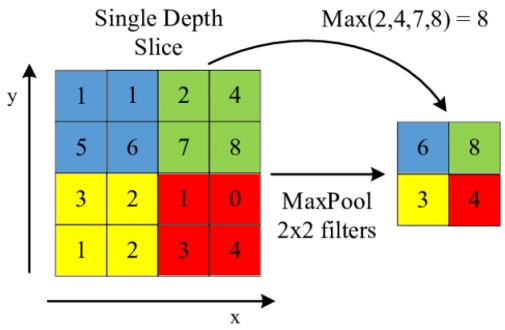

Figure 3: Max-Pooling operation (Kamencay et al., 2017)

2.2 ACTIVATION FUNCTIONS

Activation functions are a key part of neural network design (Aghdam & Heravi, 2017).The choice of activation function in the hidden layer will control how well the network model learns the training dataset. The choice of activation function in the output layer will define the type of predictions the model can make.

2.2.1 RECTIFIED LINEAR UNIT (RELU)

As seen from figure 4, ReLu is a linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.The rectified linear activation function overcomes the vanishing gradient problem, allowing models to learn faster and perform better. When developing multilayer Perceptron and convolutional neural networks the rectified linear activation is the default activation.

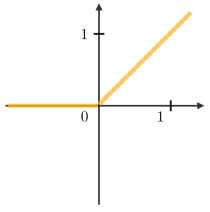

Figure 4: Rectified Linear Unit

2.2.2 SOFTMAX

The softmax step can be seen as a generalized logistic
function that takes as input a vector of scores and outputs a vector of output probability.The softmax mathematical function can be thought to be a probabilistic or "softer" version of the argmax function.Softmax can be thought of as a softened version of the argmax function that returns the index of the largest value in a list.

2.3 LOSS FUNCTIONS

Neural networks are trained using an optimization process that requires a loss function to calculate the model error. We may seek to maximize or minimize the objective function, meaning that we are searching for a candidate solution that has the highest or lowest score respectively. For multi-class classification problems cross-entropy is used as a loss function. Cross-entropy loss is minimized, where smaller values represent a better model than larger values. A model that predicts perfect probabilities has a cross entropy or log loss of 0.0.The model that is obtained has a sparse categorical cross entropy loss function that decreases over time as seen in figure 5.

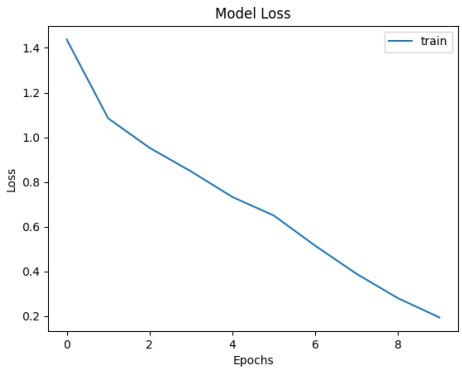

Figure 5: Model loss decrease over epoch

3 TRAINING CONVOLUTIONAL NEURAL NETWORK

Our CNN model will be trained for a few epochs and tracked its progress during training. The related hyper parameters will be chosen appropriately.

3.1 TUNING HYPERPARAMETERS

Hyperparameters determine layer architecture in the feature extraction step of a convolutional neural network (CNN), and this affects classification accuracy and learning time (Lee et al., 2018).Hyper-parameter tuning is the process of determining the right combination of hyperparameters that allows the model to maximize model performance. Setting the correct combination of hyperparameters is the only way to extract the maximum performance out of models.

3.1.1 BATCH SIZE

The number of training examples used in the estimate of the error gradient. Some types of hardware reach better run time with specific sizes of arrays. Especially when using GPUs, it is common for power of 2 batch sizes to offer better runtime. Typical power of 2 batch sizes range from 32 to 256, with 16 sometimes being attempted for large models (Goodfellow et al., 2017).The presented results confirm that using small batch sizes achieves the best training stability and generalization performance, for a given computational cost, across a wide range of experiments. In all cases the best results have been obtained with batch sizes m = 32 or smaller (Masters & Luschi, 2018).In theory, this hyper-parameter should impact training time and not so much test performance, so it can

be optimized separately of the other hyperparameters, by comparing training curves (training and validation error vs amount of training time), after the other hyper-parameters (except learning rate) have been selected.B = 32 is a good default value for starting the training (Bengio, 2012).A batch size of 32 means that 32 samples from the training dataset will be used to estimate the error gradient before the model weights are updated.In this paper batch size of 100 is used.

3.1.2 LEARNING RATE

Specifically, the learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0(Brownlee, 2020).

3.1.3 EPOCH

Number of epochs is the the number of times the entire training set pass through the neural network. We should increase the number of epochs until we see a small gap between the test error and the training error (Lau, 2017).

3.1.4 NUMBER OF LAYERS

It is usually good to add more layers until the test error no longer improves. The trade off is that it is computationally expensive to train the network. Having a small amount of units may lead to underfitting while having more units are usually not harmful with appropriate regularization.

3.2 FILTER HYPERPARAMETERS

The convolution layer contains filters for which it is important to know the meaning behind its hyperparameters.

3.2.1 FILTER SIZE

Different sized filters will detect different sized features in the input image and, in turn, will result in differently sized feature maps.It is common to use 3×3 sized filters, and perhaps 5×5 or even 7×7 sized filters, for larger input images.

3.2.2 STRIDE

For a convolutional or a pooling operation, the stride denotes the number of pixels by which the window moves after each operation.The amount of movement between applications of the filter to the input image is referred to as the stride, and it is almost always symmetrical in height and width dimensions.

3.2.3 PADDING

Zero-padding denotes the process of adding zeroes to each side of the boundaries of the input.In Keras, this is specified via the "padding" argument on the Conv2D layer, which has the default value of 'valid' (no padding). This means that the filter is applied only to valid ways to the input.The 'padding' value of 'same' calculates and adds the padding required to the input image (or feature map) to ensure that the output has the same shape as the input.

3.3 REDUCING OVERFITTING

Our neural network can consists of million or more parameters. Although, it is hard to learn so many parameters and lead to a considerable overfitting. In order to reduce overfitting, methods described below could be used.

DATA AUGMENTATION In the case of neural networks, data augmentation simply means increasing size of the data that is increasing the number of images present in the dataset. Creating more data, through data augmentation refers to randomly changing the images in ways that would not

Figure 6: Rose image with data augmentation

impact their interpretation (OV, 2019)(see figure 6). Some of the popular image augmentation techniques are flipping, translation, rotation, scaling, changing brightness, adding noise etc. (Krizhevsky et al., 2012)

4 RESULTS

The whole model is trained by using the appropriate hyper parameters. Moreover, the final model consists of multiple layers and their properties can be seen from table 1. The predictions were fairly good as seen from figure 4. The predictions for randomly selected 9 figures were accurately labeled.

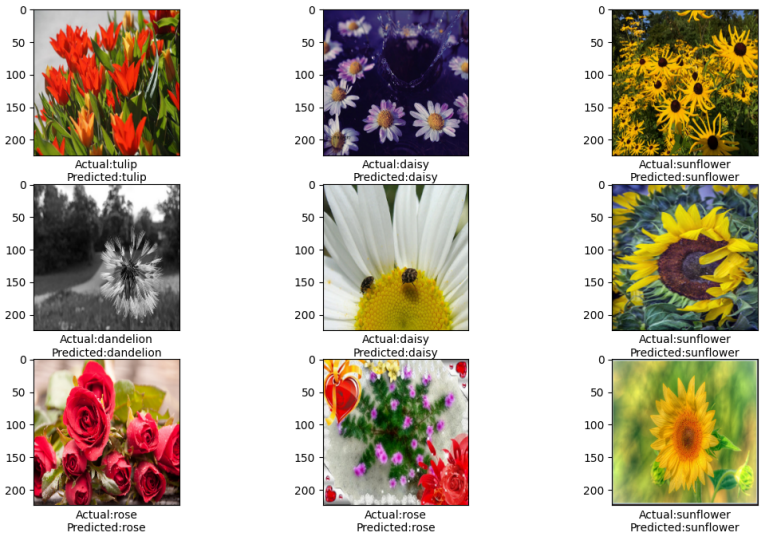

Figure 7: Actual Labels vs Predicated Labels

The confusion matrix related to this figure can be seen in table 2. Since the whole predictions are accurate the numbers appear in the diagonal.

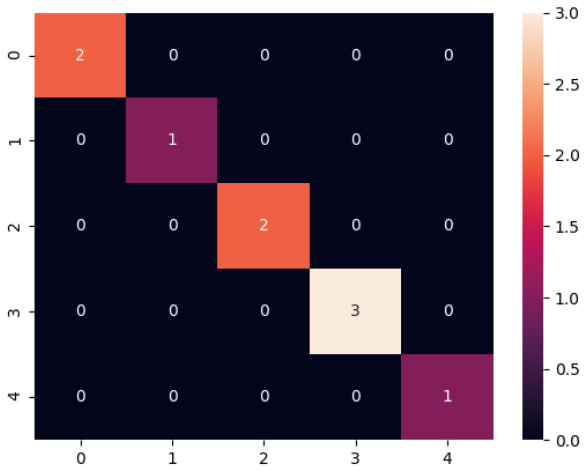

Table 2: Confusion matrix for figure 4 (0: daisy, 1: dandelion, 2: rose, 3: sunflower, 4: tulip)

5 CONCLUSION

Our final convolutional layer model consists of 11 layers with different types of layers as seen in table 1. The final CNN architecture consists of 12 layers which includes Convolution Layer, Pooling Layer, and Fully-Connected Layer. The model predicts the images with approximately 94 percent accuracy (see figure 8). Further work can be done in order to increase accuracy.

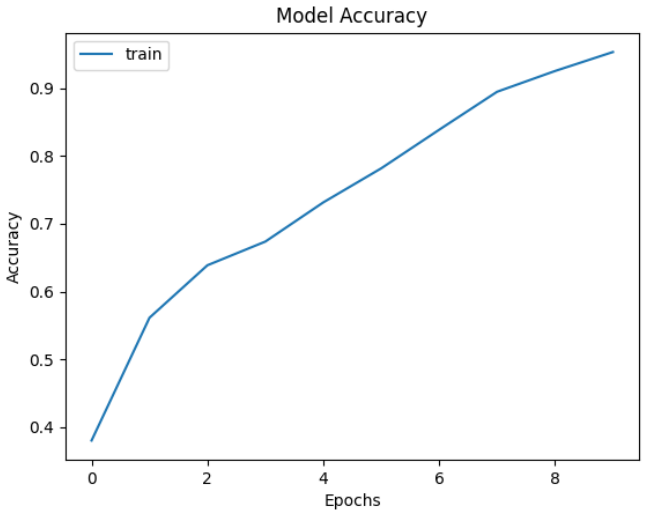

Figure 8: Model accuracy over epoch

5.1 FURTHER WORK

Before training the convolutional neural network model we may segment the flower images. Segmenting the data-set using the segmentation technique and giving the segmented data-set as the input to our CNNs can be done (Gurnani & Mavani, 2017). Different Convolutional Neural Network ar-

chitectures such as GoogleNet, AlexNet etc may be used for the classification purpose. Transfer learning proposed approach is used as an flower classification approach that reduces training time and space, which has good robustness and generalization performance (Cengil & Çinar, 2019).

REFERENCES

Convolutional neural networks cheatsheet star. URL https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks.

Hamed Habibi Aghdam and Elnaz Jahani Heravi. Guide to convolutional neural networks. 2017. doi: 10.1007/978-3-319-57550-6.

Ahmed Alsaffar, Hai Tao, and Mohammed Talab. Review of deep convolution neural network in image classification. pp. 26–31, 10 2017. doi: 10.1109/ICRAMET.2017.8253139.

Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. *Lecture Notes in Computer Science Neural Networks: Tricks of the Trade*, pp. 437–478, 2012. doi: 10.1007/978-3-642-35289-8_26.

Jason Brownlee. A gentle introduction to pooling layers for convolutional neural networks, Jul 2019. URL https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/.

Jason Brownlee. Understand the impact of learning rate on neural network performance, Sep 2020. URL https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/.

Emine Cengil and Ahmet Çinar. Multiple classification of flower images using transfer learning. pp. 1–6, 09 2019. doi: 10.1109/IDAP.2019.8875953.

dshahid380. Convolutional neural network, Feb 2019. URL https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2017.

Ayesha Gurnani and Viraj Mavani. Flower categorization using deep convolutional neural networks. 08 2017.

Patrik Kamencay, Miroslav Benco, Tomas Mizdos, and Roman Radil. A new method for face recognition using convolutional neural network. *Advances in Electrical and Electronic Engineering*, 15, 11 2017. doi: 10.15598/aeee.v15i4.2389.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012. doi: 10.1145/3065386.

Suki Lau. A walkthrough of convolutional neural network - hyperparameter tuning, Jul 2017. URL https://towardsdatascience.com/a-walkthrough-of-convolutional-neural-network-7f474f91d7bd.

Woo-Young Lee, Seung-Min Park, and Kwee-Bo Sim. Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm. *Optik*, 172: 359–367, 2018. doi: 10.1016/j.ijleo.2018.07.044.

Alexander Mamaev. Flowers recognition, Jun 2018. URL https://www.kaggle.com/alxmamaev/flowers-recognition.

Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. 04 2018.

Vladislav OV. Flower classification with convolutional neural networks., Jan 2019. URL https://towardsdatascience.com/flower-classification-with-convolutional-neural-networks-b97130329e5f.

D. Steinkraus, I. Buck, and Patrice Simard. Using gpus for machine learning algorithms. pp. 1115 – 1120 Vol. 2, 10 2005. doi: 10.1109/ICDAR.2005.251.