[[1]](#footnote-1)

*Convolutional Neural Network for LEGO Image Classification* (October 2018)

Hee Soo Yang (2016145063) and Seong Won Lee (2016145025)

*Abstract* — By training convolutional neural network (CNN), we classified 16 LEGO blocks. Training was conducted by using 22421 training images and 4484 validation images. The convolutional neural network was constructed in 41230352 parameters and three convolutional layers. For better performance of the neural network, engineering techniques such as batch normalization, max pooling, scheduling learning rate, and varying optimizer were used in convolutional layers. In this paper, four different types of neural network were compared for testing and selected as the best neural network for LEGO classifications. The best scored neural network could be regarded as the network which has the most optimal combination of variables. Especially, dropout and data augmentation were conducted for neural network to thwart overfitting.

*Index Terms*—Convolutional Neural Network, LEGO Classification, Image Augmentation. Batch Normalization, Dropout

# Introduction

A

RTIFICIAL intelligence (AI) has been developed and introduced in various fields. For example, Alpha - go has confronted a person with human based games. It is also widely used in the world’s famous companies such as Facebook, Google. Furthermore, computer vision along with autonomous vehicles and the typical drones has become one of the biggest topics related to AI research area.[3] The usage of artificial intelligence is expected to grow and is growing more actively these days. Especially, CNN[1],[2] is increasingly utilized in variety of fields such as image classification or object detection. [2] In this paper, we classified LEGO images using CNN structure in respect to their own shapes.

# The Dataset

Dataset should be categorized to train convolutional neural network. It consists of training, validation and test dataset. [8] Training set is the dataset which is used for learning. By utilizing this, parameters in neural network find optimal weights for great performance in classification. Validation set is adopted to tune the model achieving a better performance. For instance, in validation phase, validation accuracy can be our standard of when to stop training for higher accuracy. Lastly, we use training set to determine and solve out the trained model. After the model is fully-trained, it should not be trained any further.

Therefore, we have splited 31389 augmented dataset into 22421 training, 4484 validation, 4484 test images. All of the training and validation images are classified and labeled in 16 classes. Each category consists of a single type of LEGO block image viewed from various directions. Preprocessing the images enables convolutional neural network to judge the class of the LEGO block no matter which direction the LEGO block is viewed from.

# Gpu Implementation

Since the convolutional neural network has a total number of 41230352 parameters, high computing power is necessary for the training. [3] Parameters such as number of layers, stride step are varied and tested to select the best classifier. For this reason, we made advantages of GPU computing by using NVIDIA GTX 1050 Ti.

# The Architecture

Fig 1. figure illustrates our overall architecture.

## Convolution-2D

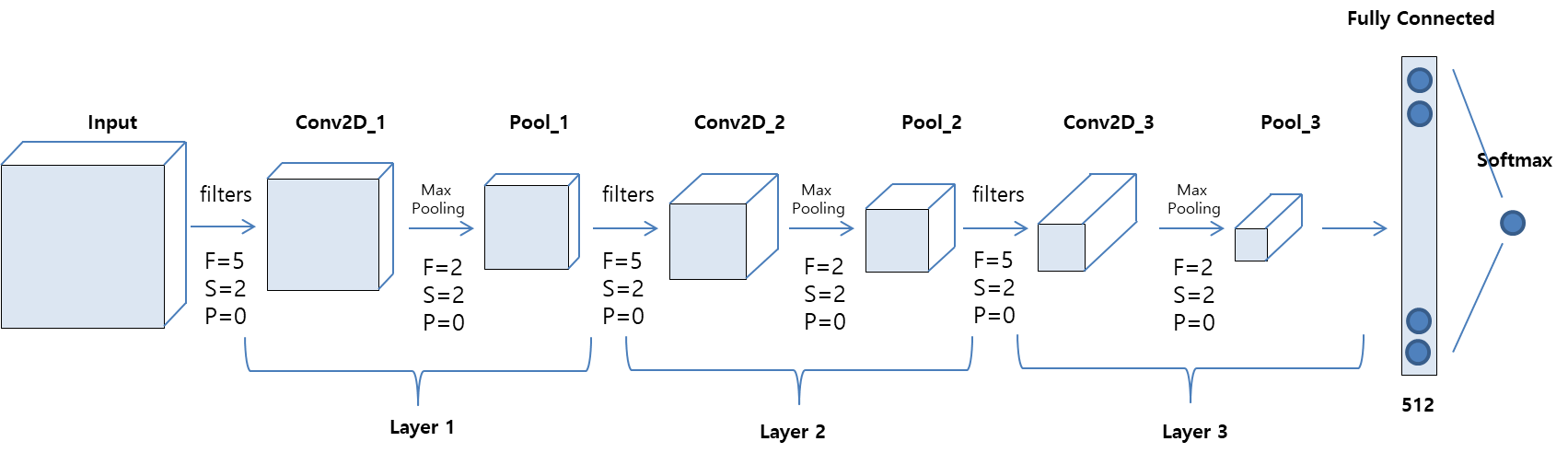
Convolution has been facilitated in image blurring, or brightness configuration. Its usages are so various that a number of technologies applied in image processing field are indirectly or directly pertain to convolution. Convolution can be divided into three factors. Element wise matrix, multiplication and sum are those. Its process is simplified in two steps: first, multiply two matrices in element wise. Second, sum of those multiplications. In a mathematical expression, it can be described as

In fact, images are multidimensional matrices that can be processed with convolution moving in different directions.

## Dense (Fully Connected)

Fully Connected refers to the type of network which all of the neuron parameters of a layer are connected. The parameters also have an operational relationship with all the other neuron parameters of the other layer.

Fig 1. Convolutional neural network architecture



## RELU

In CNN, lots of non-linear functions are employed for neuron activation. For instance, sigmoid function f(x)=1/(1 +e−x) and hyperbolic tangent function f(x) = tanh(x) are commonly observed. However, Vanishing Gradient is a critical issue for these functions. Vanishing Gradient occurs when differentiated values are so small that multiplying decimals can eventually turn them into 0. This crucial issue can be resolved with the function called RELU (Rectified Linear Unit).

## Activation (softmax)

Softmax function mathematically generalizes the logistic function and calculates the probabilities distribution over many different events. The range of the probabilities is 0 to 1 and the sum of the probabilities always equals to one. Its usage is to calculate the probability of each class in image classification with a number of classes. Activation (softmax) allows building a layer with softmax function in a neural network. This function enabled our model to represent a categorical distribution in our LEGO data.

## Pooling

Pooling is a way to sample down given data. This allows us to reduce the dimension of the data. In this convolution neural network, max-pooling technique is used. In max pooling, the technique is to divide the data into compartment which does not overlap, and to create a new dataset referring to the largest value in each compartment by using max filter.

Such a pulling technique gives a similar effect with interpreting data in a different method, and prevents overfitting. In addition, it reduces computational cost by reducing the size of data, thus enabling efficient learning

## Batch Normalization

Batch normalization is a technology to prevent Gradient Vanishing or Gradient Exploding to occur.[5] Circumventing this issue can be achieved by changing Activation function (i.e. RELU), careful initialization or small learning. However, batch normalization is an immensely direct and fundamental method to accelerate learning speed. It assumes that features are already uncorrelated. For each feature itself, means and variances are calculated in the form of scalar which will eventually be normalized. Fixed value of mean and variance such as 0 or 1 might rather remove the nonlinearity of the Activation Function. With the assumption made, the network can be limited. Batch normalization adds scale factor (gamma) and shift factor(beta) to normalized values.[5] These variables will then be trained in a similar way back-propagation does. Therefore, rather than calculating means and variances in all training data, it only calculates the variables in mini-batch which essentially improves the speed of the training process. In this process, certain values need to be defined. Input and output values are noted as follows

The algorithm is as follows

There are two well-known benefits for utilizing batch normalization. First of all, during the propagation process, batch normalization allows the process unaffected by the scale of the parameters. This phenomenon enables to set large learning rates, leading to faster training. In addition, batch normalization has its own regularization effect. This is due to the fact that the usage of dropout is equal to that of batch normalization but latter being advantageously quicker.

# Avoid Overfitting

## Augmentor

An augmentation library written in Python was used to augment our data. This library is a platform which is convenient and allows for finer control over augmentation process. It takes advantages of a stochastic approach with many different augmentation methods. The purpose of image augmentation is to artificially generate more data. This expansion of dataset as input for our deep learning model can provide a proportional amount of examples to the model, essentially enhancing fine-tuned parameters. For instance, we distorted our dataset images in order to avoid overfitting issues.[3] We also added backgrounds to some of our image datasets. This is due to the fact that the features of original dataset only include the LEGO block itself. By adding random backgrounds, our model can be utilized in general cases. (e.g. actual photo of the LEGO) Furthermore, augmentation had been done moderately so that the image data is not overly distorted.[6]

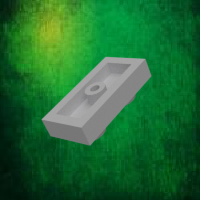
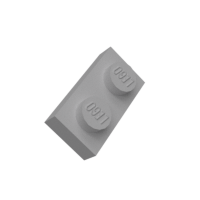


Fig 2. Augmented image data.: Original, distorted, fake background, real background.

## Dropout

Dropout is literally excluding a part of the network. Excluded network does not affect the training. Instead of building numerous models, dropout deletes some neurons randomly during the train cycle.[7] This eventually leads to training various types of models with multiple combinations of excluded neurons. The regularization effect of dropout helped our model to reduce response time.

# Optimization with Callbacks

Callback is an abstract base class facilitated to build new callbacks. This class is applied in training phase with our 2 essential functions. First function saves our weight model when the monitored value reaches the highest among previous epochs. We monitored validation accuracy to save the most accurate model while training. Second function reduces the learning rate after certain epochs. This function prevents overshooting when the initial learning rate is too high and falls into the local minimum when initial learning rate is too small. Therefore, the network can be optimally learned while downscaling from a large value to a small value.

# Experiment

|  |  |  |
| --- | --- | --- |
|  | Adam | SGD |
| With BatchNorm | 99.3 0.2% | 94.7 0.4% |
| Without BatchNorm | 93.0 0.5% | 80.1 1.0% |

Our model was trained with two different kind of optimizers; Adam and SGD. We also differentiated our model by adding or removing batch normalization. Each model was trained with 50 epochs at least three times to ensure accurate data. Epochs were continuously configured with our callback function which schedules learning rate for every epoch. After training, model was examined with evaluation and the log was recorded. The figure shows that our most accurate model involves Batch Normalization and Adam optimizer. The accuracy of each model was measured with internal Keras library.

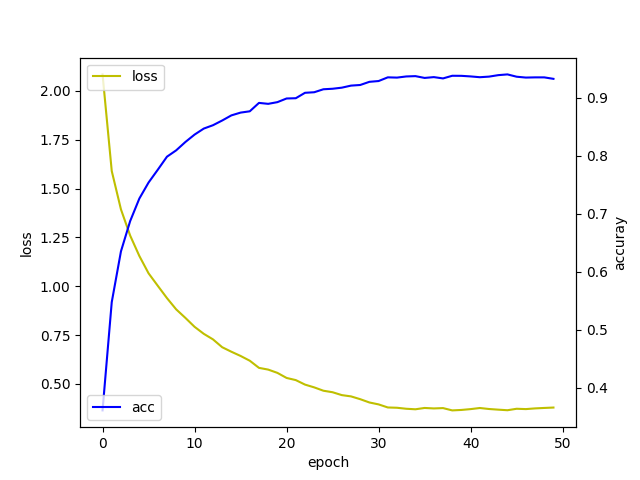
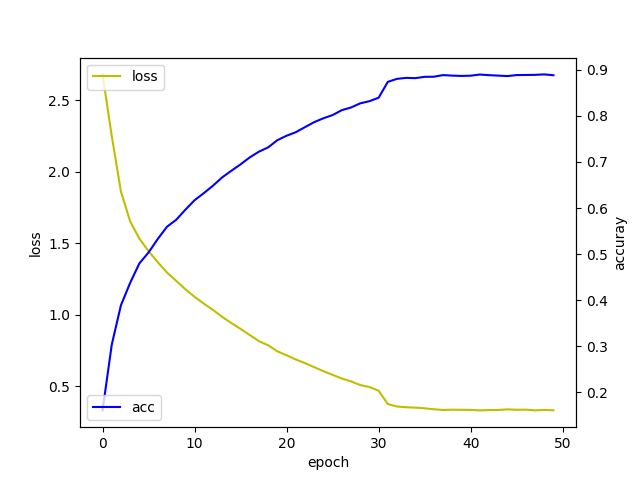
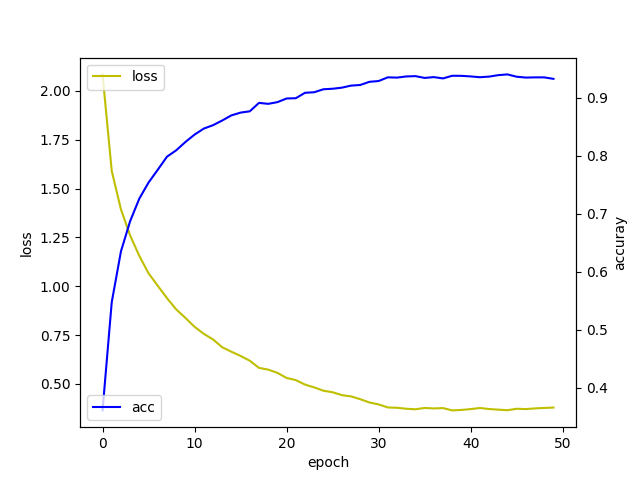
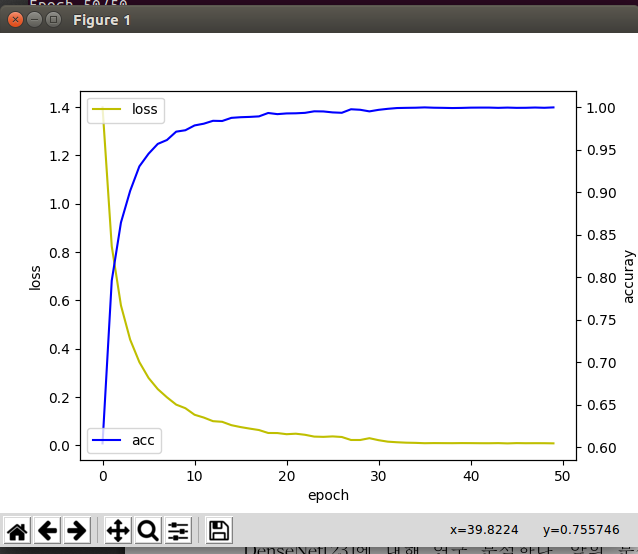


Fig 3. Training loss and accuracy logs

The figure above illustrates the loss and accuracy of our final model with their accuracy. We also have tested our model with each picture of every LEGO classes. This was especially helpful to visualize our model prediction. In other words, the prediction error in certain classes could be easily detected and further modified. For instance, the results of the experiments are as follows.

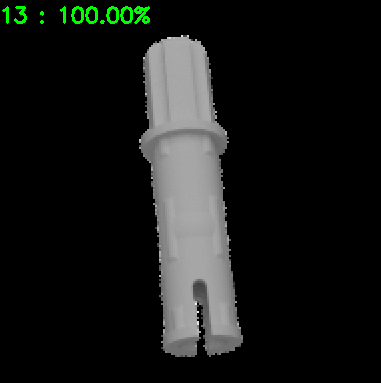
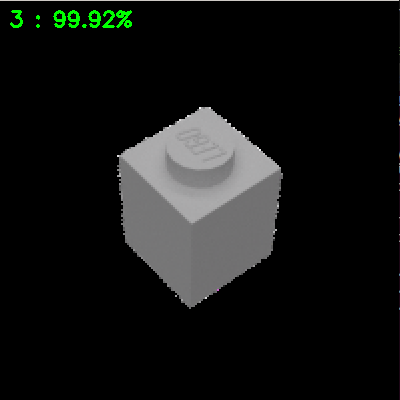
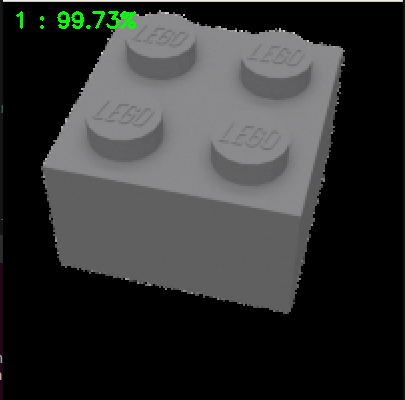
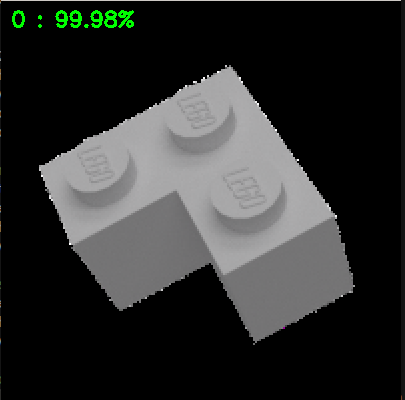


Fig 4. Accuracy results of a single image

# Conclusion

Experimental results show that the convolutional neural network is 99% accurate in our network. The accuracy was found to be significantly different according to which optimizer and the learning rate we choose. The learning rate,and the optimum network was obtained accordingly.

To summarize our experiments, the convolutional neural network was implemented with dropout and trained with augmented images to prevent overfitting. In addition, testing with arbitrary LEGO images resulted in an accuracy of 99 percent. We expect additional testing is required to show better performance, such as ensemble learning or transfer learning as a method to prevent overfitting and increase accuracy.

Ensemble Learning is a technique that applies a variety of machine learning techniques in a variety of ways by mixing various models, in order to reduce the variance and bias. [6],[7] In addition, transfer learning is a technique for learning new dataset using pre-trained weights, which is a method used to take advantage of previously learned networks.[6],[7] With such other methods, better performance could be obtained.

References

1. Lawrence, Steve, et al. "Face recognition: A convolutional neural-network approach." *IEEE transactions on neural networks* 8.1 (1997): 98-113.
2. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.
3. Hoo-Chang, Shin, et al. "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning." *IEEE transactions on medical imaging* 35.5 (2016): 1285.
4. Oh, Kyoung-Su, and Keechul Jung. "GPU implementation of neural networks." *Pattern Recognition* 37.6 (2004): 1311-1314.
5. Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).
6. Howard, Andrew G. "Some improvements on deep convolutional neural network based image classification." *arXiv preprint arXiv:1312.5402* (2013).
7. Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *The Journal of Machine Learning Research* 15.1 (2014): 1929-1958.
8. Krogh, Anders, and Jesper Vedelsby. "Neural network ensembles, cross validation, and active learning." *Advances in neural information processing systems*. 1995.
9. Hansen, Lars Kai, and Peter Salamon. "Neural network ensembles." *IEEE transactions on pattern analysis and machine intelligence* 12.10 (1990): 993-1001.
10. Oquab, Maxime, et al. "Learning and transferring mid-level image representations using convolutional neural networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.

1. [↑](#footnote-ref-1)