УДК 004.852
МРНТИ 28.23.25

MAX-POOL AND DROPOUT REGULARIZATION DEEP LEARNING TECHNIQUES TO DETECT TRAFFIC SIGNS

A. YEREZHEPBEKOV

International Information Technology University

Abstract: *many car drivers are inattentive to traffic signs which result in unfortunate or even dramatic accidents, so in order to prevent such things this article proposes using machine learning technique convolutional neural networks with max-pool and dropout regularization algorithms. Recently, a dropout regularization technique has seen increasing use in deep learning. For deep convolutional neural networks, dropout is known to work well in fully-connected layers. However, its effect in convolutional and pooling layers is still not clear. This article illustrates in pythonic manner that max-pooling dropout is equivalent to randomly picking activation based on a multinomial distribution at training time. Training set is implemented upon a famous German traffic sign dataset and to see the difference between two regularization methods. Since, dropout regularizer is very efficient in minimizing the overfitting of the training set by randomly discarding inbound and outbound neurons. Plus, in mix with max-pooling a dropout regularization might require more epochs to converge more accurately. Feeding the algorithm with traffic sign dataset makes it useful for adaptive cruise control systems in cars to avoid nasty and awkward car accidents. Two methods can be used in tandem or separately but in either case performance can be tuned by changing hyperparameters.*

Keywords: *Deep learning; Convolutional neural networks; Max-pooling dropout*

МАКС-ПУЛ МЕН “DROPOUT” ТЕРЕҢ ОҚЫТУ ӘДІСТЕМЕСІН РЕТТЕУ ТӘСІЛДЕРІН ҚОЛДАНУ АРҚЫЛЫ ЖОЛ БЕЛГІЛЕРІН АНЫҚТАУ

Аңдатпа: *көптеген автокөлік жүргізушілері жол белгілеріне назар аудармайды, соның нәтижесінде олар бақытсыз немесе тіпті қатерлі апаттарға әкеп соғады, бұлай болдырмаудың алдын алу үшін машинада оқыту әдістемесін нейрондық желілерді “max-pool” және “dropout” рекортизациялау алгоритмдерімен пайдалану ұсынылады. Жақында “dropout” регламенттеу әдістемесі терең білім алуда қолданудың артықшылығын көрсетті. Терең конвектуралық нейрондық желілер үшін, толықтай жалғанған қабаттарда тастау тиімді жұмыс істейді. Алайда, конвалитациялық және топырақты қабаттарға әсері әлі күнге дейін толық зерттелмеген. Бұл мақала “python” тәсілмен суреттеледі, ол максималды біріктіріліп шығуы жаттығу уақытында мультиномиальды үлестіруге негізделген кездейсоқ жинақтауды белсендіруге тең. Оқу жиынтығы әйгілі неміс жол белгісі деректер жиынтығымен орындалады және екі регламенттеу әдісі арасындағы айырмашылықты көреді. Өйткені, үзіліс регистраторы жаттығу жиынтығын кіріс және шығыс нейрондарды кездейсоқ алып тастау арқылы азайтуға өте ыңғайлы. Сонымен қатар, максималды бірліктермен араласқанда, кетуді регуляризациялау дәлірек жақындау үшін көп кезеңді қажет етеді. Алгоритмді қозғалыс белгісінің деректер жиынтығымен азықтандыру оны көліктердегі бейімделгіш круиздік басқару жүйелеріне ыңғайсыз етеді, бұл жолсыз және ыңғайсыз көлік оқиғаларын болдырмайды. Екі әдісті тандемде немесе бөлек қолдануға болады, бірақ екі жағдайда да гиперпараметрлерді өзгерту арқылы орындалуы мүмкін.*

Түйінді сөздер: *терең меңгеру, тұрақты нейрондық желілер, максималды-біріктіргіш шығару, регуляризация, Байес теориясы, жаттықтыру*

МЕТОДЫ РЕГУЛЯРИЗАЦИИ ГЛУБОКОГО ОБУЧЕНИЯ MAX-POOL И DROPOUT ДЛЯ ОБНАРУЖЕНИЯ ДОРОЖНЫХ ЗНАКОВ

Аннотация: *Многие водители автомобилей невнимательны к дорожным знакам, которые приводят к несчастным или даже драматическим случаям. Поэтому, чтобы предотвратить такие вещи, в этой статье предлагается использовать технику машинного обучения сверточными нейронными сетями с алгоритмами максимального пула и повторного отсева. В последнее время методика регуляризации отсева находит все большее применение в глубоком обучении. Известно, что для глубоко сверточных нейронных сетей отсеивание хорошо работает в полностью связанных слоях. Однако его влияние на сверточный и объединяющий слои все еще неясно. В этой статье наглядно показано, что отсев максимального пула эквивалентен случайному выбору активации на основе полиномиального распределения во время обучения. Учебный комплект реализован на основе известного немецкого набора данных дорожных знаков и позволяет увидеть разницу между двумя методами регуляризации, поскольку регуляризатор отсева очень эффективен для минимизации переобучения обучающего набора путем случайного отбрасывания входящих и исходящих нейронов. Кроме того, в сочетании с максимальным пулированием для регуляризации отсева может потребоваться больше эпох, чтобы более точно сходиться. Заполнение алгоритма набором данных дорожных знаков делает его полезным для адаптивных систем круиз-контроля в автомобилях, чтобы избежать неприятных и неуклюжих автомобильных аварий. Два метода могут использоваться в тандеме или по отдельности, но в любом случае производительность может быть настроена путем изменения гиперпараметров.*

Ключевые слова: *глубокое обучение, сверточные нейронные сети, макс-пул отбрасывание, регуляризация, теория Байеса, тренировка*

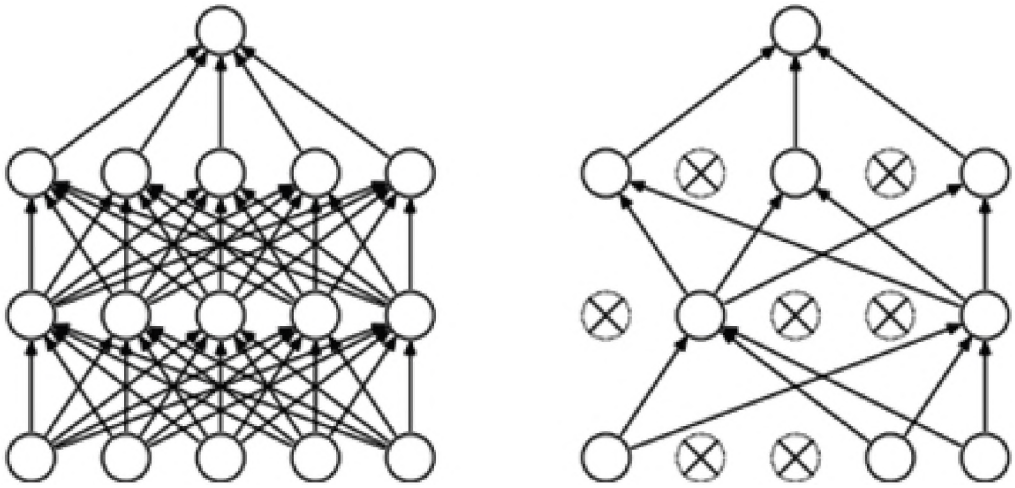
1 Introduction

CNN (Convolutional Neural Networks) - is an useful part of deep neural networks which has made a huge success in 1997 when it was first introduced by Yann LeCunn. Due to its requirement for a large amount of computational power which was unavailable at that time the method was forgotten for several years until recently with the advent of GPU (Graphical Processing Unit) and a Dropout regularization approach.

Model mix almost usually improves the performance of machine learning methods. With large neural networks, however, an apparent solution of averaging the outputs of many separately trained networks is computationally expensive. Combining several models is helpful when the individual models are different from each other they should either have different architectures or be trained on different data. Training many different architectures is hard because finding optimal hyperparameters for each architecture is almost impossible feat to perform and training each large network is computationally exhaustive. Plus, large networks normally require large amounts of training data and there may not be enough data available to train different networks on different subsets of the data. Even if one was able to train many different large networks, using them all at test time is infeasible in applications where it is important to respond quickly.

Dropout (Hinton et al., 2012) is a recently proposed regularizer to fight against over-fitting. It is a regularization method that stochastically sets to zero the activations of hidden units for each training case at training time. This breaks up co-adaptions of feature detectors since the dropped-out units cannot influence other retained units. Another way to interpret dropout is that it yields a very efficient form of model averaging where the number of trained models is exponential in that of units, and these models share the same parameters. Dropout has also inspired other stochastic model averaging methods such as stochastic pooling (Zeiler & Fergus, 2013) and DropConnect (Wan et al., 2013).

Dropout is a method that prevents over-fitting and provides a way of approximately combining exponentially many different neural network models efficiently. The term “dropout” refers to dropping out units (hidden and visible) in a neural network. By removing a unit, we mean temporarily dropping it out from the network, along with all its inbound and outbound



a) Standard neural net *b) After dropout*

Figure 1.1 - Dropout Neural Net Model. Left: A standard neural net with 2 hidden layers. Right: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

connections, as shown in Figure 1. The choice of which units to drop is random. In the simplest case, each unit is retained with a fixed probability p independent of other units, where p can be chosen using a validation set or can simply be set at 0.5, which seems to be close to optimal for a wide range of networks and tasks. For the input units, however, the optimal probability of retention is usually closer to 1 than to 0.5.

2 Dropout regularization against traditional CNN in other researches

CNNs have far been known to produce remarkable performance on MNIST (LeCun et al., 1998), but they, together with other neural network models, fell out of favor in practical machine learning as simpler models such as SVMs became the popular choices in the 1990s and 2000s. With deep learning renaissance (Hinton & Salakhutdinov, 2006; Ciresan, Meier, & Schmidhuber, 2012; Bengio, Courville, & Vincent, 2013), CNNs regained attentions from machine learning and computer vision community. Like other deep models, many issues can arise with deep CNNs if they are naively trained. Two main issues are computation time and over-fitting. Regarding the former problem, GPUs help a lot by speeding up computation significantly.

To combat over-fitting, a wide range of regularization techniques have been developed. A simple but effective method is adding *l*2 penalty to the network weights. Other common forms of regularization include early stopping, Bayesian fitting (Mackay, 1995), weight elimination (Ledoux & Talagrand, 1991) and data augmentation. In practice, employing these techniques when training big neural networks provides better test performances than smaller networks trained without any regularization.

Dropout is a new regularization technique that has been more recently employed in deep learning. It is similar to bagging (Breiman, 1996), in which a set of models are trained on different subsets of the same training data. At test time, different models' predictions are averaged together. In traditional bagging, each model has independent parameters, and all members would be trained explicitly. In the case of dropout training, there are exponentially many possibly trained models, and these models share the same parameters, but not all of them are explicitly trained. Actually, the number of explicitly trained models is not larger than me, where m is the number of training example, and e is the training epochs. This is much smaller than the number of possibly trained models, (n is number of hidden units in a feed-forward neural networks). Therefore, a vast majority of models are not explicitly trained at training time.

At test time, bagging makes a prediction by averaging together all the sub-models' predictions with the arithmetic mean, but it is not obvious how to do so with the exponentially many

models trained by dropout. Fortunately, the average prediction of exponentially many sub-models can be approximately computed simply by running the whole network with the weights scaled by retaining probability. The approximation has been mathematically characterized for linear and sigmoidal networks (Baldi & Sadowski, 2014; Wager el al., 2013); for piecewise linear networks such as rectified linear networks, Warde et al. (2014) empirically showed that weight-scaling approximation is a remarkable and accurate surrogate for the true geometric mean, by comparing against the true average in small enough networks that the exact computation is tractable.

Since dropout was thought to be far less advantageous in convolutional layers, pioneering work by Hinton et al. (2012) only applied it to fully-connected layers. It was the reason they provided that the convolutional shared-filter architecture was a drastic reduction in the number of parameters and thus reduced its possibility to overfit in convolutional layers. Wonderful work by Krizhevsky et al. (2012) trained a very big convolutional neural net, which had 60 million parameters, to classify 1.2 million high-resolution images of ImageNet into the 1000 different categories. Two primary methods were used to reduce over-fitting in their experiments. The first one was data augmentation, an easiest and most commonly used approach to reduce over-fitting for image data. Dropout was exactly the second one. Also, it was only used in fully-connected layers. In the ILSVRC-2012 competition, their deep convolutional neural net yielded top-5 test error rate of 15.3%, far better than the second-best entry, 26.2%, achieved by shallow learning with hand-craft feature engineering. This was considered as a breakthrough in computer vision. From then on, the community believes that deep convolutional nets not only perform best on simple hand-written digits, but also really work on complex natural images.

Compared to original work on dropout, (Srivastava et al., 2014) provided more exhaustive experimental results. In their experiments on CIFAR-10, using dropout in fully-connected layers reduced the test error from 15.60% to 14.32%. Adding dropout to convolutional layers further reduced the error to 12.61%, revealing that applying dropout to convolutional layers aided generalization. Similar performance gains can be observed on CIFAR-100 and SVHN. Still, they did not explore max-pooling dropout.

Stochastic pooling (Zeiler & Fergus, 2013) is a dropout-inspired regularization method. The authors replaced the conventional deterministic pooling operations with a stochastic procedure. Instead of always capturing the strongest activity within each pooling region as max-pooling does, stochastic pooling randomly picks the activations according to a multinomial distribution. At test time, probability weighting is used as an estimate to the average over all possible models. Interestingly, stochastic pooling resembles the case of using dropout in max-pooling layers, so it is worth comparing them.

3 Using max pooling in combination with dropout

A traffic sign classifier is a combination of two popular regularization techniques which are convolutional max-pooling and convolutional dropout. The former is used to pick the maximum value from a kernel from the previous layer, which means it chooses a channel with higher intensity. Like in the figure 2 the straight line is not classified because 0 i.e. black color can't be picked up by a max pooling. Therefore, it has such disadvantage of losing some valuable information along the way. But if we apply it to a different picture with a switched background color in figure 3. In that case max pooling performs the best possible prediction almost identical to the original picture.

Now there is a possibility of taking an advantage of this feature of max-pooling and patch it up with dropout. A standard CNN consists of convolutional and pooling layers, with fully-connected layers on the top and on each presentation of a training example, if layer *l* is followed by a pooling layer, the forward propagation without dropout can be described as follows:

$$a_j^{(l+1)} = F_{(pool)}\left(a_1^{(l)}, \dots, a_i^{(l)}, \dots, a_n^l\right), i \in R_j^l \quad (3.1)$$

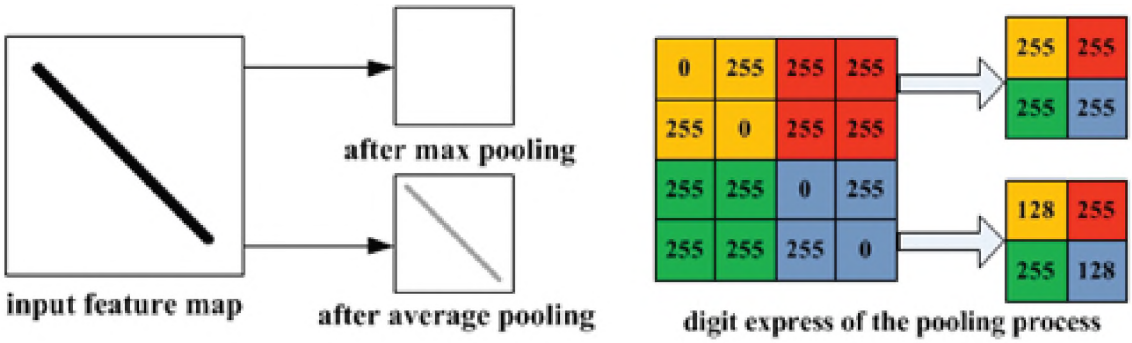


Figure 2. Illustration of a max pooling disadvantage

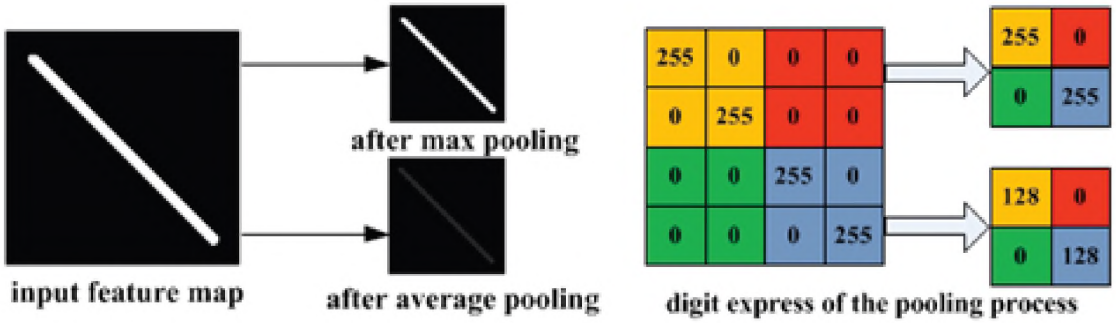


Figure 3.1 – Illustration of an average pooling disadvantage

$a_j^{(l+1)}$ – an activation unit

$F_{(pool)}$ – pool function

R_j^l – pooling region *j* at layer *l*

Here is R_j^l pooling region j at layer l and $a_j^{(l+1)}$ is the activation of each neuron within it. n = R_j^l is the number of units in R_j^l. $F_{(pool)}$ denotes the pooling function. Pooling operation provides a form of spatial transformation invariance as well as reduces the computational complexity for upper layers. An ideal pooling method is expected to preserve task-related information while discarding irrelevant image details. Two popular choices are average-and max-pooling. Average-pooling takes all activations in a pooling region into consideration with equal contributions. This may downplay high activations as many low activations are averagely included. Max-pooling only captures the strongest activation, and disregards all other units in the pooling region. We now show that employing dropout in max-pooling layers avoids both disadvantages by introducing stochasticity.

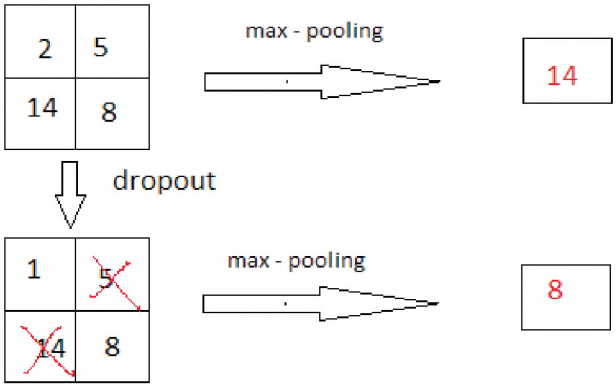


Figure 3.2 – An illustrating example of the procedure of max-pooling dropout. The activation in the pooling region is 2, 5, 14 and 8 respectively. Without dropout, the strongest activation6is always selected as the output. With dropout, each unit in the pooling region could be possibly eliminated. In this example, only 1 and 8 remained, then 8 will be the pooled output.

4 Implementation of the max-pooling and dropout in python

There is an amazing machine learning package called Tensorflow available in python [1]. So, first of all we used a famous German traffic signs dataset to retrieve images for feeding to our algorithm. Second of all, there is LeNet convolutional algorithm, initially created

by Yann Lecun [5]. He used MNIST dataset to recognize numbers from zero to ten and that input data was millions of handwritten numbers which were even illegible for human eyes. We took a part of that algorithm and transformed it to recognize traffic signs.

The full dataset consisted of 51,839 images RGB images with dimensions 32x32. 34,799 images were used as the training dataset, 12,630 images were used as the testing dataset, and 4,410 images were used as the validation dataset.

A validation set was used to assess how well the model is performing. A low accuracy on the training and validation sets implies underfitting. A high accuracy on the training set but low accuracy on the validation set implies overfitting. The validation set was purely used to calibrate the network's hyperparameters.

In total, the dataset consisted of images belonging to 43 classes. Each class corresponds to a specific sign, for example, the class with label 4 represents 70km/h speed limit signs, and the class with label 25 represents a roadwork sign.

A sample from each class is shown in the image below:

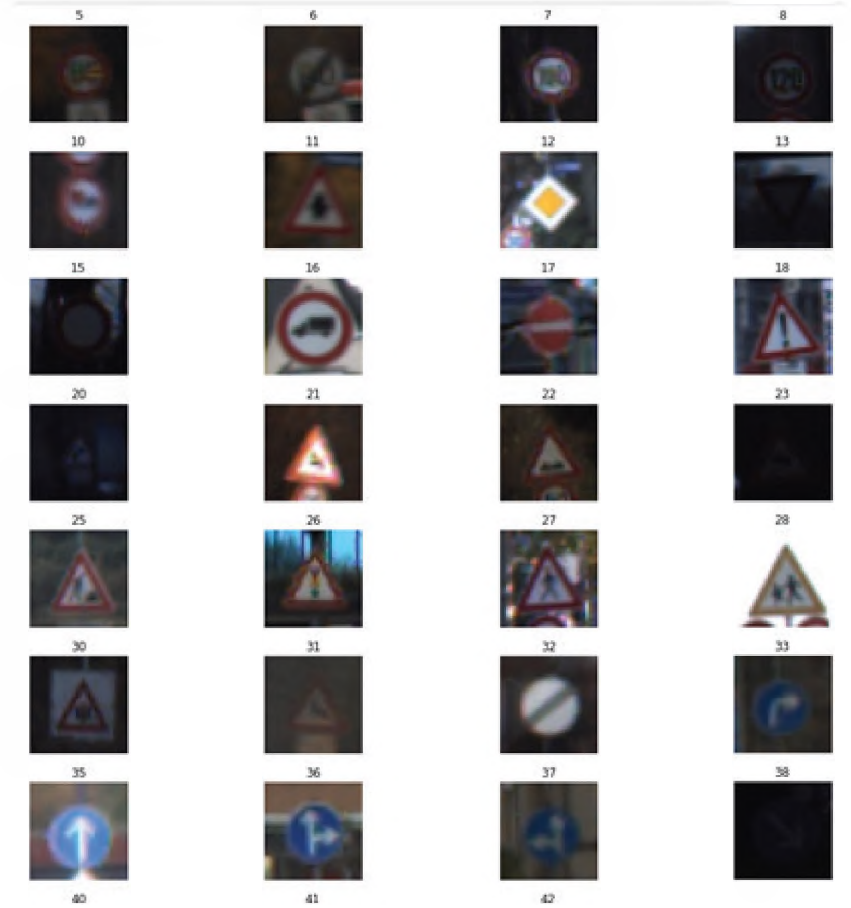


Figure 4.1 – Traffic sign dataset

The pixel data of each image was normalized, and then fed into the Drop_Max neural network which consisted of the following layers. At the first stage the data is normalized. Each image is 3 channel 32x32x3 RGB which are fed to the input of Convolution 5x5 (1x1 stride, valid padding and 28x28x26 output neurons). After that a new reshaped matrix of pixels go through ReLu which then move to Max pooling (2x2 stride, 16x16x6 outputs) and again fed into Convolution activation (1x1 stride, valid padding, 10x10x16). In each step a number of parameters has been decreasing which denotes that the network drops out repetitive weights that clutter the implementation. For example at this current stage there are 1600 parameters and that number will decrease. After the second Convolution the output neurons inbound to ReLu activation function after it they

are pooled and finally all the output 3 dimensional matrix is flattened.

But now we added *dropout* activation function which is included in TensorFlow library. Here *dropout* takes what is outbound from max-pooling as inputs and produce its own neurons for further units. In droput function we should indicate a percentage of units being removed from the layer and we picked 0.75 as an optimum parameter.

Below there are two implementations which better show the result:

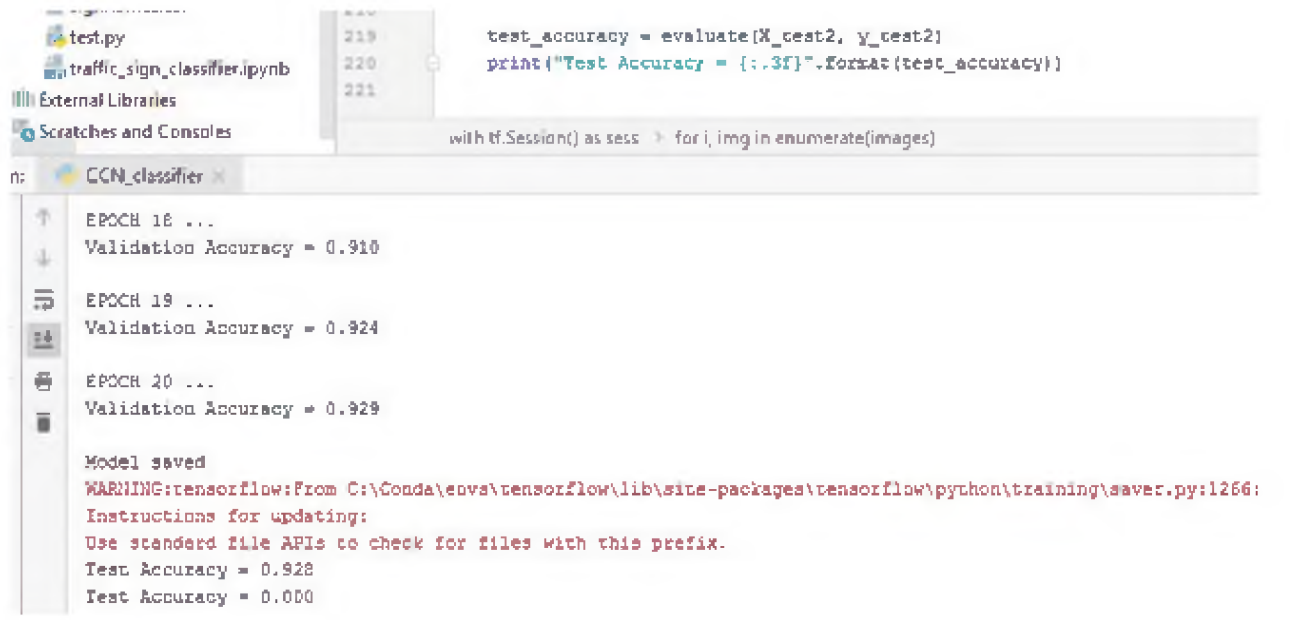


Figure 4.2 – max-pooling implemented in python with an accuracy 0.929 and epoch 20

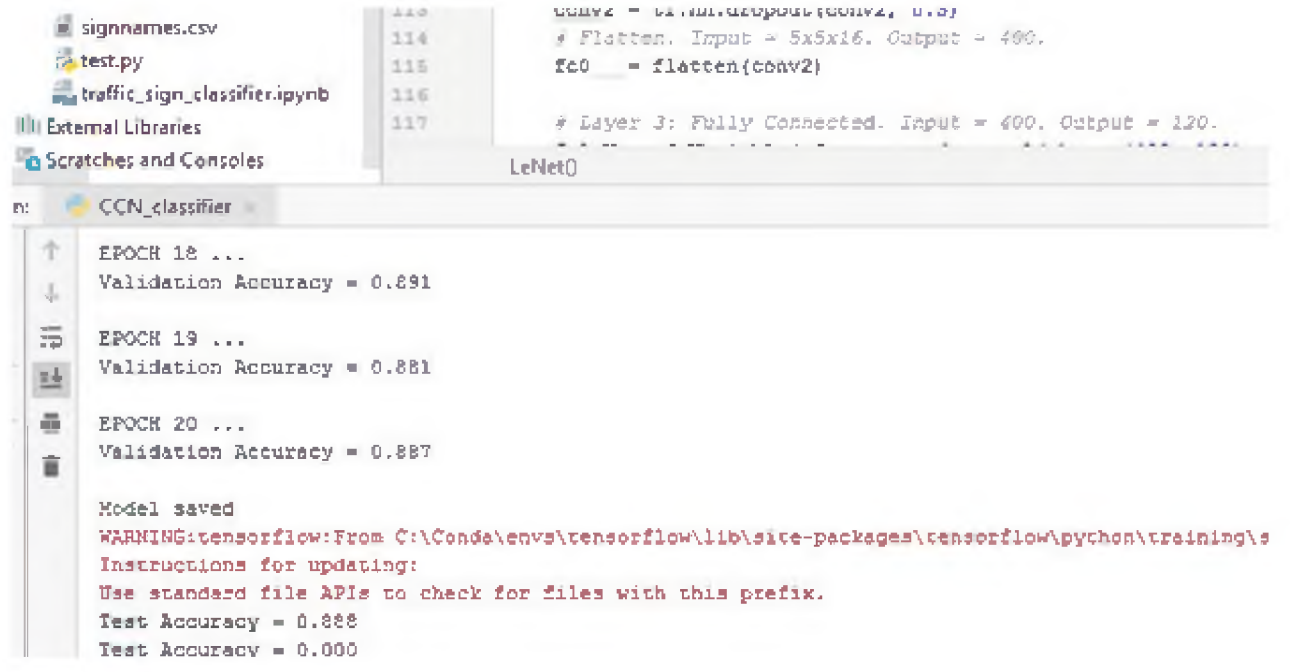


Figure 4.3 – max-pooling combined with dropout with the accuracy 0.888 and epoch 20

So according to the result from the code above, when max-pooling is used alone there is a higher accuracy than if it is combined with dropout activation. However, even the accuracy of dropout is lower there is a possibility to play with a hyperparameters and get as high accuracy as possible.

Conclusion

The article mainly addresses the problem of using max-pool with dropout regularization in order to better understand the difference between the two. Due to the low variance of input distribution of traffic sign dataset images and good weight sharing techniques of convolutional neural networks processing time took us only several minutes. With higher epochs accuracy gets better but for the sake of experiment there is only 20 epochs which is enough to show the satisfied result. We also have done many experiments, unfortunately the scope of which is beyond the content of the article, but we have seen that both methods perform well with perfectly tuned hyperparameters. There is also no data augmentation method used which is also good for computational speed and such as algorithm with dropout regularization can be applied in collision avoidance systems for cars.

REFERENCES

1. Baldi, P., & Sadowski, P. (2014). The dropout learning algorithm. *Artificial Intelligence*, 210, 78-122.
2. Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: a review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35, 1798-1828.
3. Boureau, Y. L., Ponce J., & LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *Proceedings 27th of International Conference on Machine Learning (ICML 2010)*.
4. Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24, 123-140.
5. Ciresan. D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *Proceedings of 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2012)*.
6. Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013). Maxout networks. In *Proceedings of 30th International Conference on Machine Learning (ICML 2013)*.
7. Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313, 504-507.
8. Hinton, G. E., Srivastave, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaption of feature detectors. *arXiv 1207.0580*.
9. Springenberg J. T., & Riedmiller M. (2014). Improving deep neural networks with probabilistic maxout units. In *Proceedings of 3rd International Conference on Learning Representations (ICLR 2014)*.
10. Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. *M.S. diss., Department of Computer Science, University of Toronto*.
11. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS 2012)*.
12. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*.
13. Ledoux, M., & Talagrand, M. (1991). Probability in banach spaces. *Springer*.
14. Lin, M., Chen, Q., & Yan S. (2014). Network in network. In *Proceedings of 3rd International Conference on Learning Representations (ICLR 2014)*.
15. Mackay, D. C. (1995). Probable networks and plausible predictions: A review of practical Bayesian methods for supervised neural networks. In *Bayesian Methods for Backpropagation Networks*.
16. Scherer, D., Muller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *Proceedings of 20th International Conference on Artificial Neural Networks (ICANN 2010)*.

17. Srivastava, N., Hinton. G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15, 1929-1958.
18. Vinod, N., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings 27th of International Conference on Machine Learning (ICML 2010)*.
19. Wan, L., Zeiler, M. D., Zhang, S., LeCun, Y., & Fergus, R. (2013). Regularization of neural networks using DropConnect. In *Proceedings of 30th International Conference on Machine Learning (ICML 2013)*.
20. Warde, F. D., Goodfellow, I.J., Courville, A., &Bengio, Y. (2014). An empirical analysis of dropout in piecewise linear networks. *In Proceedings of 3rd International Conference on Learning Representations (ICLR 2014)*.
21. Wager, S., Wang, S., &Liang, P. (2013). Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems (NIPS 2013)*.
22. Zeiler, M. D., & Fergus R. (2013). Stochastic pooling for regularization of deep convolutional neural networks. In *Proceedings of 2nd International Conference on Learning Representations (ICLR 2013)*.