# Dropblock

## Neural Network

Sapienza Università di Roma

Ludovico Ottobre, 1712005
Gianmarco Evangelista, 1711818

June 7, 2021

**Abstract** – Regularization is a technique used to reduce the errors by fitting the function appropriately on the given training set and avoid overfitting. It refers to a model that models the training data too well. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data.
There have been several methodologies put forward for regularization for instance L1 and L2 regularization, Dropout, Data augmentation, Early Stopping and Data Augmentation. Dropout is widely used as a regularization technique for fully connected layers but it is often less effective for convolutional layers.

DropBlock, a form of structured Dropout, improves the performances of Dropout especially in deep architectures with skip connections where skip some layer in the neural network and feed the output of one layer as the input to the next layers where units in a contiguous region of a feature map are dropped together. As we will see, it improves accuracy in image classification and results to be a good choice in object detection.

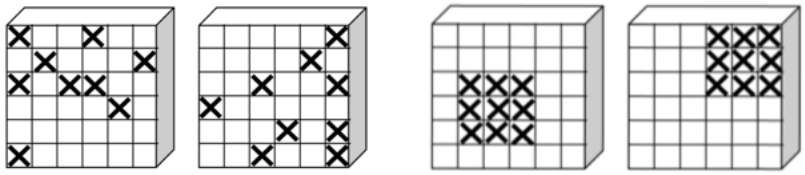
# 1 Introduction

We have decided to reimplement *DropBlock*, a regularization method like *Dropout, SpatialDropout* where contiguous region of a feature map, are dropped together.

Overfitting refers to the phenomenon where a neural network models the training data very well but fails when it sees new data from the same problem domain. It is caused by noise in the training data that the neural network picks up during training and learns it as an underlying concept of the data. Overfitting causes the neural network model to perform very well during training, but the performance gets much worse during inference time when faced with brand new data.

Regularization refers to a set of different techniques that lower the complexity of a neural network model during training, and thus prevent the overfitting. DropBlock method was introduced to combat the major drawback of Dropout being dropping features randomly which prove to be effective strategy for fully connected networks but less fruitful when it comes to convolutional layers wherein features are spatially correlated. DropBlock technique discards features in a contiguous correlated area called block. As DropBlock discards features in a correlated area, the networks must look elsewhere for evidence to fit the data.

In our experiments, we have used DropBlock in a range of models and datasets. Adding DropBlock to specific architecture, we have improved image classification accuracy.



(a) Dropout (b) DropBlock

## 2 DropBlock

Dropout has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs. DropBlock method was introduced to resolve the problem of Dropout in which each features are dropped randomly while in DropBlock we have that this technque discards features in a contiguous correlated area called block. For this reason, this method has the capability to satisfy the idea to do a more simple model that allow us to improve the algorithm since we reduce overfitting and we achieve better performances.

The principal parameters for DropBlock algorithm are the size of the block to be dropped i.e. *blocksize* and how many activation units to be dropped i.e. $\gamma$ with each feature channel having its DropBlock mask.

**Algorithm 1** DropBlock

**Input:**output activations of a layer ($A$), $block\_size$, $\gamma$, $mode$
**if** $mode == Inference$ **then**
    return $A$
**end if**
Randomly sample mask $M$: $M_{i,j} \sim Bernoulli(\gamma)$
For each zero position $M_{i,j}$, create a spatial square mask with the center being $M_{i,j}$, the width, height being $block\_size$ and set all the values of $M$ in the square to be zero
Apply the mask: $A = A \times M$
Normalize the features: $A = A \times \textbf{count}(M)/\textbf{count\_ones}(M)$

Following the above pseudocode, we have reimplemented DropBlock in Keras. We set a different value for *block_size* in order to obtain the highest verification set accuracy. In particular, for each experiment, we have used two different fixed value: 3 and 7. We have done this choice to evaluate if changing the block_size value, we have different results. The second main parameter is $\gamma$. It is a float between 0 and 1 and it controls how many activation units to drop. where:

- *keep_prob* is the threshold probability set wherein all the elements whose probability is less than keep_prob are effectively removed;
- *feat_size* is the size of feature map;
- *feat_size - block_size + 1* is the valid seed region.

Following the idea proposed in the reference paper, we implemented a callback function that allowed us to decrease the keep_prob value over time. We use a linear scheme to decrease the value in question by a certain amount for each epoch.

```
[ ]  class myCallback(keras.callbacks.Callback):
        def on_epoch_end(self, epoch,logs={}):
            for i in range(31): #number of DropBlock layer in the model
              s = str(i)        #string in order to recognize layer
              # vary the rate wrt epochs
              self.model.get_layer(name = s).keep_prob -= 0.07
```

Figure 1: Callback function

```
history = model.fit(x_train, y_train,
              batch_size=batch_size,
              epochs=epochs,
              validation_data=(x_test, y_test),
              shuffle=True,
              callbacks = myCallback())
```

Figure 2: Calling Callback function in model.fit

# 3 Dataset

Differently from original paper, in the project we developed we have used two of the most commonly used datasets, namely MNIST and CIFAR-10 which are described more in detail in the following subsections. We chose to evaluate these datasets since the dataset used in the original paper was too large to manage and would not have allowed us to reach any result with the means we have available.

## 3.1 CIFAR-10

The CIFAR-10 dataset is composed by 60,000 of 32x32 colour images from 10 classes, such as airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. We have chosen to have 50,000 images for the training dataset and 10,000 for the test set. In the following we can see some images taken from the dataset.

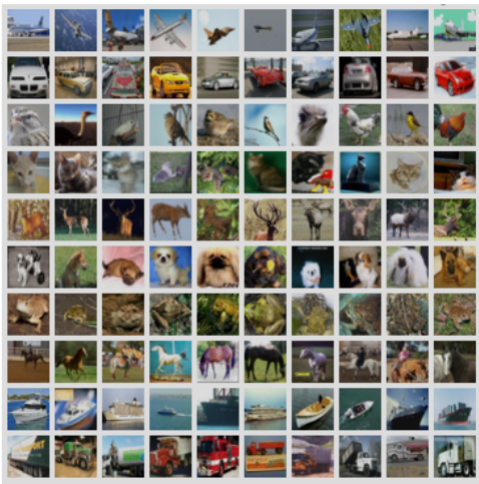


Figure 3: CIFAR-10 structure

### 3.2 MNIST

The MINST dataset is composed by 70,000 small square 28x28 gray scale images that contains single handwritten integer number between 0 to 9, divided as 60,000 images for training and 10,000 images for testing. In the following we can see some images taken from the dataset.

Figure 4: MNIST structure

# 4 Experiments

In order to evaluate the correctness and functionality of *DropBlock*, we decided to use three different neural networks: sequential CNN, *DenseNet* and *ResNet*. The networks were chosen to be able to analyze and compare the results obtained with those that were already available to us. Each of the networks was processed in five different ways: without the use of regularization methods, with *Dropout*, with *SpatialDropout*, with *DropBlock* (block_size = 3) and with DropBlock (block_size = 7). Since the DropBlock, as suggested by the paper under analysis, are useful and efficient especially for networks with skip connection, in addition to considering ResNet, we have chosen DenseNet for our analysis. In sequential CNN, as we will see later, DropBlock does not guarantee improvements over Dropout.

**Dropout**. The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1 / (1 - rate) such that the sum over all inputs is unchanged.

**SpatialDropout2D**. It performs the same function as Dropout, however, it drops entire 2D feature maps instead of individual elements. If adjacent pixels within feature maps are strongly correlated (as is normally the case in early convolution layers) then regular dropout will not regularize the activations and will otherwise just result in an effective learning rate decrease. In this case, SpatialDropout2D will help promote independence between feature maps and should be used instead.

In order to proof the efficiency and limits of DropBlock we decided, as previously mentioned, to carry out our analyzes on different neural networks and compare the results with the other regularization methods. We trained our models on Colab using GPU and using the implementations of Res-Net, DenseNet and a custom CNN. Since baselines are usually overfitted for the longer training scheme and have lower validation accuracy at the end of training, we report the highest accuracy over the full training course for fair comparison. We have used a different number of epochs for each model based on the time needed and Colab limits.

### 4.1.1 DropBlock in ResNet

ResNet-50 is a CNN architecture widely used for image recognition. We have applied different regularization techniques on ResNet-50 in order to compare the result with DropBlock.

| MODEL | Accuracy CIFAR-10 | Accuracy MNIST |
|---|---|---|
| ResNet-50 | 0.6878 | 0.9900 |
| ResNet-50 + *Dropout(0.25)* | 0.6619 | 0.9920 |
| ResNet-50 + *SpatialDropout(0.7)* | 0.5477 | 0.9807 |
| ResNet-50 + *DropBlock(bs = 3, kp = 0.9)* | 0.6935 | 0.9898 |
| ResNet-50 + *DropBlock(bs = 7, kp = 0.9)* | 0.6743 | 0.9876 |

Figure 5: ResNet-50 accuracy

The configuration we chose to be able to insert DropBlock inside the model was to put the regularization method at the end of each *Conv - Batch - Activation* block, then immediately after the *Relu* activation function. The output of a building block is the sum of outputs from the convolutional branch and skip connection.

In order to do a comparison among all the chosen methods, we have considered also the original ResNet. We applied DropBlock with *block_size = 3 and block_size = 7*. We decreased $\gamma$ for each epoch, inserting inside the fit method, a callback function in order to manage this value during the training. In all configurations, DropBlock and dropout share the similar trend and DropBlock has a large gain compared to the best dropout result. This shows how DropBlock is a more effective regularizer compared to Dropout. In the same way, even comparing SpatialDropout and DropBlock, we can see that there is an important difference between the two methods.

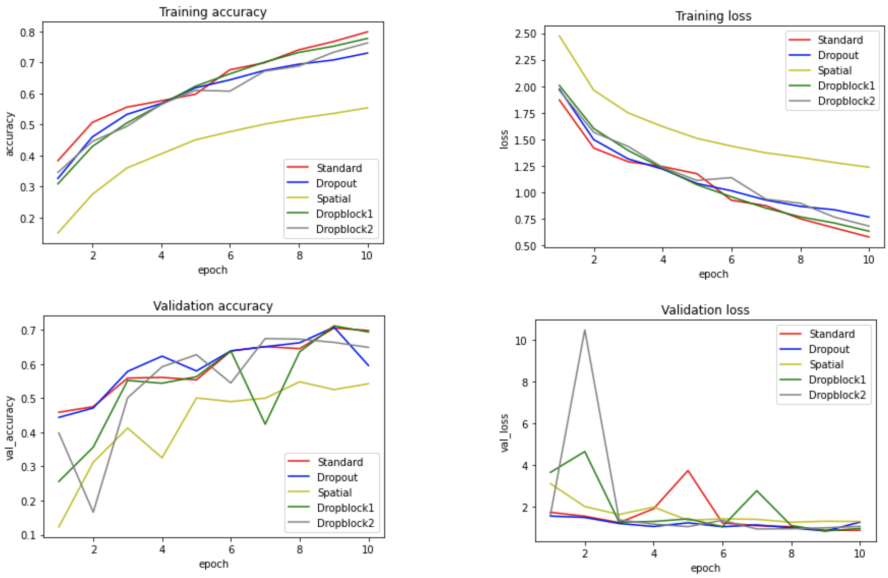



Figure 6: ResNet-50 performances on CIFAR-10

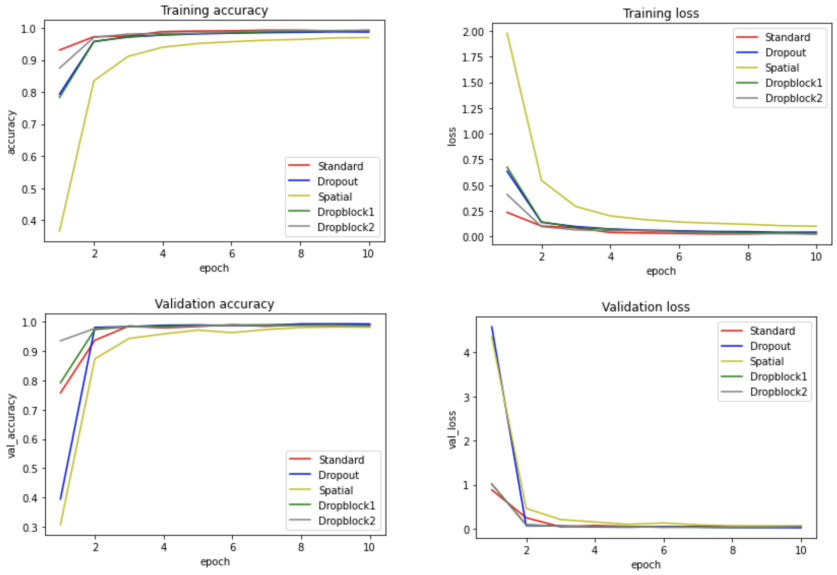



Figure 7: ResNet-50 performances on MNIST

### 4.1.2 DropBlock in DenseNet

DenseNet is quite similar to ResNet with some fundamental differences. ResNet uses an additive method that merges the previous layer (identity) with the future layer, whereas DenseNet concatenates the output of the previous layer with the future layer. It results to be, like ResNet, a model with skip connection so we have chosen to analyze DropBlock on this Net since we have the same situation of ResNet. By using a skip connection, we provide an alternative path for the gradient (with backpropagation). It is experimentally validated that this additional paths are often beneficial for the model convergence. Skip connections in deep architectures, as the name suggests, skip some layer in the neural network and feeds the output of one layer as the input to the next layers (instead of only the next one).

| MODEL | Accuracy CIFAR-10 | Accuracy MNIST |
|---|---|---|
| DenseNet | 0.6581 | 0.9899 |
| DenseNet + *Dropout(0.25)* | 0.6060 | 0.9893 |
| DenseNet + *SpatialDropout(0.3)* | 0.6105 | 0.9799 |
| DenseNet + *DropBlock(bs = 3, kp = 0.9)* | 0.7040 | 0.9910 |
| DenseNet-50 + *DropBlock(bs = 7, kp = 0.9)* | 0.7146 | 0.9919 |

Figure 8: DenseNet accuracy

The configuration we have chosen to be able to insert DropBlock within DenseNet, follows the one that was the choice made in the case of ResNet in order to be able to evaluate the trend of the regularization method in two different models but in the same situation and therefore in the same position. Again, we inserted DropBlock at the end of each *Conv - Batch - Activation* block, so right after the *Relu* activation function.

In order to do a comparison among all the chosen methods, we have considered also the original DenseNet. We applied DropBlock with *block_size = 3 and block_size = 7*. We decreased $\gamma$ for each epoch, inserting inside the fit method, a callback function in order to manage this value during the training. Differently from ResNet, the results in this case seem to be stronger. With DropBlock method we achieve a better accuracy (+5%).

In this configuration, DropBlock and dropout don't share a similar trend and DropBlock has a large gain compared to the best dropout result. This shows how DropBlock is a more effective regularizer compared to Dropout. In the same way, even comparing SpatialDropout and DropBlock, we can see that there is an important difference between the two methods.
An important aspect, seeing the graphs reported below, is the robustness of Dropblock which it seems to achieve the better result on different evaluation parameters.

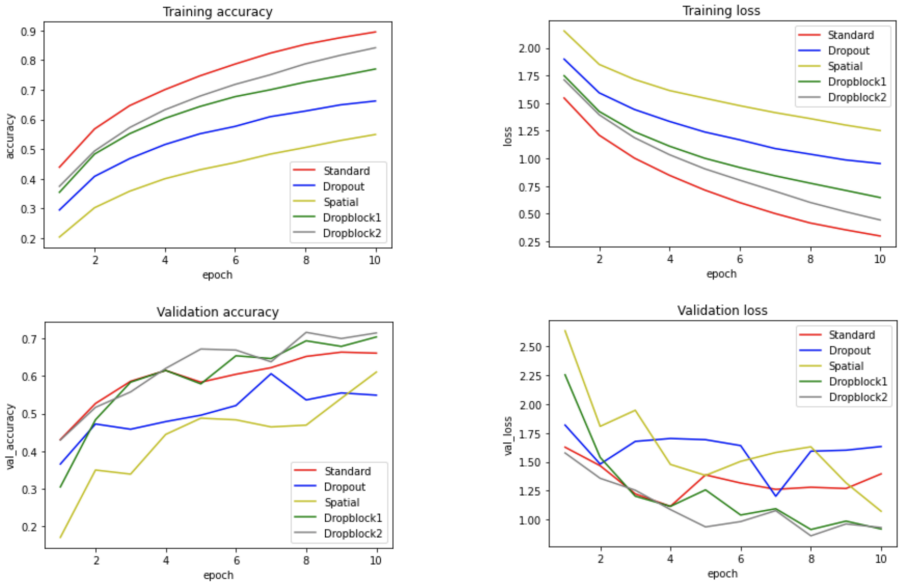



Figure 9: DenseNet performances on CIFAR-10

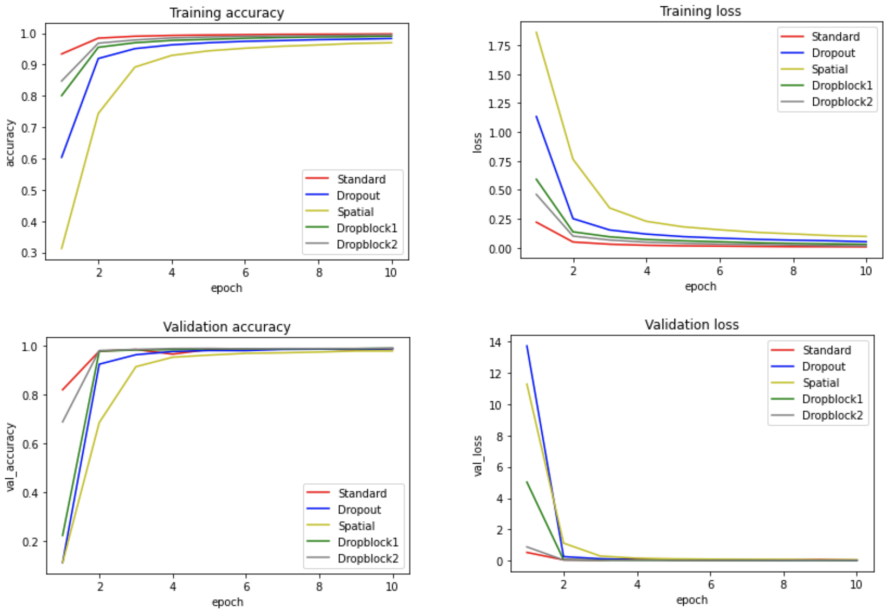



Figure 10: DenseNet performances on MNIST

### 4.1.3 DropBlock in a CustomModel

This CNN has be done in order to evaluate DropBlock in a different architecture no skip connection. In this model we have a lower number of parameters compared to the architectures analyzed previously.

| MODEL | Accuracy CIFAR-10 | Accuracy MNIST |
|---|---|---|
| Custom Model | 0.7813 | 0.9939 |
| Custom Model + *Dropout(0.25)* | 0.8264 | 0.9945 |
| Custom Model + *SpatialDropout(0.7)* | 0.7617 | 0.9941 |
| Custom Model + *DropBlock(bs = 3, kp = 0.9)* | 0.8004 | 0.9931 |
| Custom Model + *DropBlock(bs = 7, kp = 0.9)* | 0.7956 | 0.9945 |

Figure 11: CustomNet accuracy

In this case we have inserted Dropblock after *MaxPooling2D* since it was the best position in order to obtain the better result. In this model we have a number of epochs equal to 100 so we have a more gradual decrease of *keep_prob*.

Also in this case DropBlock behaves in a good way but Dropout can be achieve the best results for accuracy and loss. We have done this last experiment in order to show how DropBlock performances change if we don't have a model with skip connection. Although, as we can see in the figures below, dropout is the best choice, Dropblock still achieves good results compared to SpatialDropout and the model without any kind of regularization method.

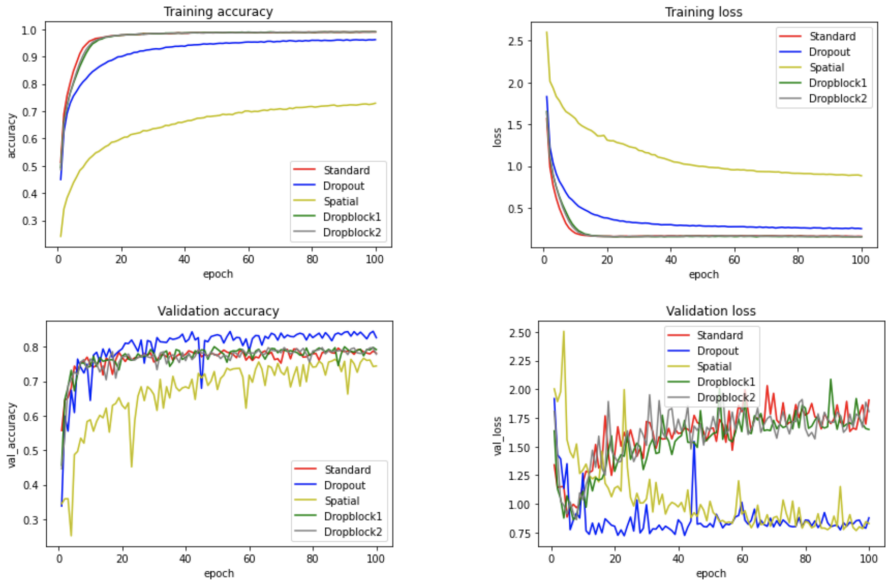



Figure 12: CustomNet performances on CIFAR-10

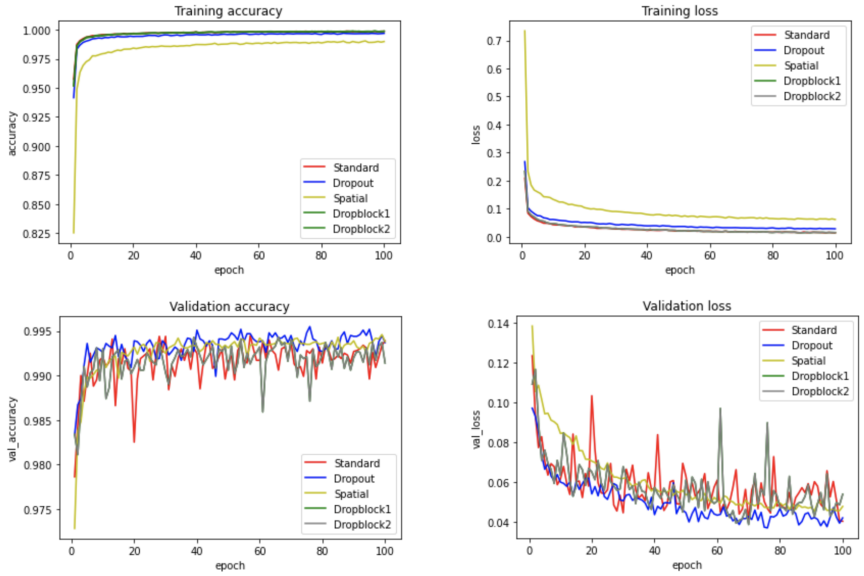



Figure 13: CustomNet performances on MNIST

## 4.2 Experimental Analysis

DropBlock proves to achieve better accuracy than the other regularization methods used and generally improves what is the official implementation of ResNet50 and DenseNet.
The idea behind DropBlock working properly and thus getting good results is based on the idea that units in a contiguous region of a feature map are put together. We have seen that applying DropBlock to jump connections in addition to convolution levels increases accuracy.

Also with MNIST dataset, where we have a more simple image respect to CIFAR-10 and so quite different from the images present in ImageNet, we can achieve good performances even if there are better regularization methods.

## 4.3 Object Detection

Object detection is a computer vision technique that allows us to identify and locate objects in an image or video. With this kind of identification and localization, object detection can be used to count objects in a scene and determine and track their precise locations, all while accurately labeling them.
In order to implement and evaluate DropBlock performance also in this computer vision technique, we have used RetinaNet based on ResNet-50 and we have inserted inside ResNet-50 FPN the DropBlock method. Using as dataset COCO, we can see even in this case how the behavior of DropBlock with respect to the official implementation, where we don't have any regularization method, results to be quite well.

COCO dataset is a large-scale object detection, segmentation, and captioning dataset. We have used a smaller subset of 500 images for training in this case.

As we have already said, we have included the DropBlock method inside the ResNet FPN. RetinaNet uses a ResNet based backbone, using which a feature pyramid network is constructed. In order to make this change, we applied DropBlock on *tf.nn.relu(output)*. We tried DropBlock with different values for *block_size* and *keep_prob*. In general we have decided to evaluate RetinaNet considering *loss* and *val_loss*.

We have decided to make a comparison with the Standard implementation, Dropout and SpatialDropout. As we can see from the the following table, even in this case DropBlock results to be a valid option in order to make Object Detection. Analyzing the results based on *loss* and *val_loss*, we can say that lower is the loss, better is the model. The loss is calculated on training and validation and its interpretation is how well the model is doing for these two sets. Unlike accuracy, loss is not a percentage. It is a summation of the errors made for each example in training or validation sets. Loss value implies how well or poorly a certain model behaves after each iteration of optimization. Ideally, one would

expect the reduction of loss after each, or several, iteration(s). We considered the lowest loss value for each model and, making a theoretical evaluation of the results, we can see how DropBlock turns out to be a good alternative to other regularization methods such as Spatial and Dropout.

| MODEL | loss | val_loss |
|---|---|---|
| RetinaNet | 1.6236 | 2.6221 |
| RetinaNet + *Dropout(0.25)* | 1.6299 | 2.5462 |
| RetinaNet + *SpatialDropout(0.7)* | 1.8380 | 2.6808 |
| RetinaNet + *DropBlock(bs = 7, kp = 0.7)* | 1.7043 | 2.5539 |
| RetinaNet + *DropBlock(bs = 7, kp = 0.9)* | 1.6841 | 2.5540 |
| RetinaNet + *DropBlock(bs = 3, kp = 0.7)* | 1.6414 | 2.6322 |
| RetinaNet + *DropBlock(bs = 3, kp = 0.9)* | 1.6985 | 2.6010 |

Figure 14: RetinaNet on COCO2017

# 5 Conclusion

The DropBlock technique been evidenced to effectively beat some of the best performance results obtained by traditional Dropout and Spatial Dropout as well as shows better performance corresponding to strong data augmentation techniques. DropBlock has proved to be effective regularization approach for object detection. Our experiments show that applying DropBlock in skip connections in addition to the convolution. It is a form of structured dropout that drops spatially correlated information.

We demonstrate DropBlock is a more effective regularizer compared to Dropout

in CIFAR-10 and MNIST classification, in the same way results to be a good alternative for object detection in COCO dataset.

# 6 References

[1] Ghiasi, Golnaz, Tsung-Yi Lin, and Quoc V. Le. "Dropblock: A regularization method for convolutional networks." Advances in Neural Information Processing Systems. 2018.

[2] `https://keras.io`

[3] `https://keras.io/examples/vision/retinanet/`

[4] `https://github.com/tensorflow/tpu/tree/master/models/official/resnet`

[5] `https://github.com/taki0112/Densenet-Tensorflow`