Self-supervised ConvNet model for learning semantic features

Kevin Dalla Torre
kevindt@kth.se

Laura Cros
lcros@kth.se

Sena Soyleyici
senas@kth.se

Abstract

This project covers the re-implementation of the paper "Unsupervised Representation Learning by Predicting Image Rotations"[1], in which the Network in Network (NIN) architecture[3] was used for training. In the paper, a self-supervised learning approach was proposed in which the images were rotated (0°, 90°,180°, 270°) and used for rotation prediction during training, to extract semantic features without requiring manual annotation. Those learned features were then used by means of transfer learning to classify images. CIFAR-10 dataset was used. In addition to this paper we experimented with a network that used the learned features to predict the bounding box of traffic signs in images. The network and algorithm were implemented in Keras. The results of the experiments showed that the self-supervised approach to extracting features does help to obtain a good representation of the images that can be used in other tasks such as classification or regression. The code can be found on this link: `https://github.com/kevindallatorre/DD2424_project_group_55.git`

1 Introduction

In the Machine Learning field, one of the main problems is acquiring the amount of labeled data required to train a model. When it comes to image processing, the main goal is to learn the representations that best define the images, although it requires a great amount of manual labeling. By using a self-supervised feature learning approach like the one used here, manual labeling is not required to extract representative features. In order to do so, a Convolutional Neural Network was trained to predict the rotation applied to an image. The labels used were generated while training (self-supervised approach) representing each rotation.

After training the network, the performance of the self-supervised features is evaluated on the image classification task for CIFAR-10 dataset. Although we wanted to go a little further, and see if the features extracted were representative enough to perform a completely different task such as detecting traffic signs and predicting the position of the bounding box, by using the German Traffic Sign Dataset[4].

2 Related Work

The paper called "Unsupervised Representation Learning by Predicting Image Rotations" was chosen for our project. We implemented some experiments that are mentioned in that paper, mainly based on the proposed Rotational Network (RotNet) and Network in Network (NIN) architectures. State of art classification performance on CIFAR-10 is achieved in the paper using the RotNet and the NIN architectures. In the paper, the experiments were performed in an unsupervised manner. We followed their steps to re-implement the network by ourselves and we used the same parameters that they used. After that, we trained the networks. They implemented their networks on Pytorch, ours are implemented on Keras having some functions re-implemented by us. Apart from trying to reproduce the same results they got, we tested how good the representations learned by the RotNet were. We did that by implementing a regression task to predict the bounding box of traffic signs in images, employing transfer learning to make use of the representations extracted from the RotNet.

3 Data

For our project, we used two different datasets. The first one and the main one is the CIFAR-10 dataset[2]. It has 10 different labels within 50000 images for training and 10000 images for testing. When training the self-supervised network, we rotated the images in 4 degrees (0°, 90°,180°, 270°), meaning that while training the network sees 4 times more images, so in total, we used 200000 images for training and 40000 images for testing. The same dataset was used for the subsequent classification tasks, using the CIFAR-10 classes as targets which are 10.

The second dataset we used is The German Traffic Sign Detection Benchmark[1]. It has 900 images in total. For the regression task, we pre-processed the images to fit into our model, as the image sizes vary between 15x15 to 250x250 pixels. In order to do so, we first resize the images 4 times smaller. After that, we randomly crop them around the bounding box of the traffic sign in the image to have size 32x32. The coordinates of the new bounding boxes were saved in a vector containing: the x coordinate of the top-left corner of the traffic sign bounding box, the y coordinate of that point, the width and the height of the bounding box. In total, the number of images in the new dataset was 3639, which were randomly split 85% for training and 15% for testing.

4 Methods

The main point of our project is to test how well does self-supervised learning work for extracting representative features from images. To do so, we use the self-supervised RotNet. This architecture consists of 4 blocks that have three convolutional networks in them (see Figure 2). After training, we took the first two convolutional blocks from the network, froze them, and added more layers for classification and regression purposes. In this section we discuss the different architectures we used.

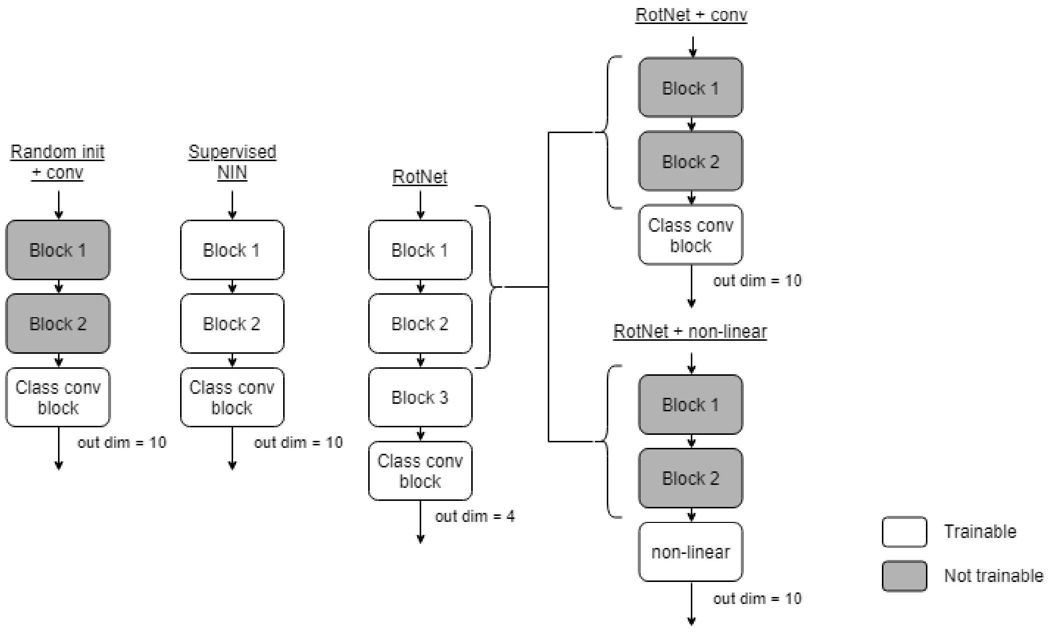


Figure 1: Diagram of different architectures used

[1]http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset

4.1 Network in Network (NIN)

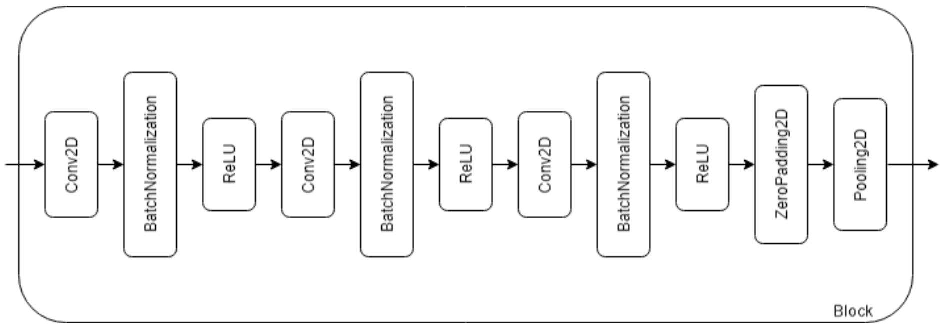


Figure 2: NIN- one block

Convolutional Neural Networks have different types of structure. NIN[3] is one type of them. It consists on sequentially stacking different layers of convolutional networks to create blocks, having a last layer of pooling. The diagram of the basic block we used can be seen in Figure 2, in which we used batch normalization and ReLu activation function after each convolutional layer. The type of pooling of the block depends on the situation of the block in the global network. If it is the first block, then the pooling used is max pooling. If it is the last block, then it is global average pooling. For the other blocks, the pooling used is average pooling.

Global average pooling is used as the last layer in order to enforce correspondence between feature maps and categories[3].

4.2 Rotational Network (RotNet)

RotNet is a network that uses self-supervised learning to extract representative features of images without needing labelled images. The way this self-supervised approach works is that, Before training, we rotate the images and create a label according to that rotation. The network then is trained to classify the input images (rotated versions). When doing so, the convolutional blocks learn to extract meaningful features of the images.

Our version of RotNet uses 4 different degrees of rotation: 0°, 90°,180° and 270°. In Figure 3 we can see the four different rotations applied to the input images.

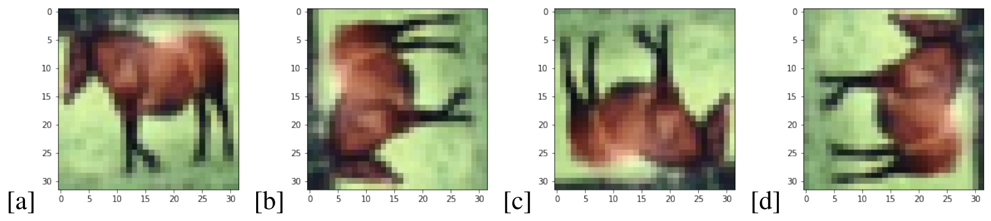


Figure 3: (a) 0 degrees (y=0) (b) 90 degrees (y=1) (c) 180 degrees (y=2) (d) 270 degrees (y=3)

After generating the rotated images and the new labels, we train those images with stacked blocks, following the previously mentioned NIN architecture. In our experiments, the RotNet consists on 4 blocks. The last layer of the last block is a fully connected layer to classify the images depending on the rotation. The diagram of the architecture can be found in Figure 1.

4.3 Classification Network

For the classification task, we wanted to follow two different approaches to see which one works better. For both we used the first two blocks of the pre-trained RotNet, froze them and added either fully-connected layers or a convolutional block. Both methods classify the images in 10 different labels.

4.3.1 RotNet + non-linear

The non-linear block consists on three fully connected layers, stacked after the two RotNet blocks. For this approach, after the two blocks we added a flatten layer in order to have a 1-dimensional feature vector. After that, two fully connected layers with 200 units were added, which had batch normalization and ReLu activation function. The last layer is another fully connected layer of 10 units with softmax activation function to classify the 10 labels.

4.3.2 RotNet + Conv

In this architecture, we add another convolutional block. This block is defined as the last convolutional block we described in the NIN section, having as last layer a global average pooling. Following that layer, we stacked a fully connected layer of 10 units with softmax activation function to classify the 10 labels.

4.4 Traffic Sign Regression

In this part we tried different approaches, but the one which gave us the best results is a combination of non-linear and convolutional blocks. After extracting and freezing the first two blocks of the RotNet, we added another convolutional block which had, instead of a pooling layer, a flatten layer as its last layer. Following that block, we added two fully connected layers of 200 nodes with batch normalization and ReLu activation function. Lastly, we stacked a fully connected layer of 4 units without any activation function, to give the four values that define the bounding box.

5 Experiments

We did different experiments with different settings. Firstly, we trained the RotNet model to later use it by means of transfer learning for the classification and regression tasks. We tested which block is the one that gives the best feature map and, after choosing it, we used that part of the RotNet to perform experiments for classification and regression.

We trained all the following experiments for 100 epochs, with 0.9 momentum, 5e-4 weight decay.

5.1 RotNet

In the following table we can see the specifications of the convolutional layers of the four blocks of the RotNet:

Table 1: The feature map and kernel size of first two Conv blocks

	Block 1			Block 2		
Conv layer	1^{st}	2^{nd}	3^{rd}	1^{st}	2^{nd}	3^{rd}
Feature Map	192x32x32	96x32x32	160x32x32	192x16x16	192x16x16	192x16x16
Kernel Size	5	1	1	5	1	1

Table 2: The feature map and kernel size of last two Conv blocks

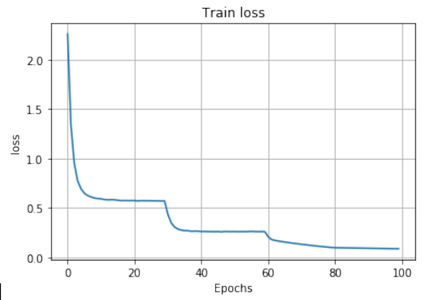
	Block 3			Class Conv Block		
Conv layer	1^{st}	2^{nd}	3^{rd}	1^{st}	2^{nd}	3^{rd}
Feature Map	192x8x8	192x8x8	192x8x8	192x4x4	192x4x4	192x4x4
Kernel Size	3	1	1	3	1	1

The pooling layers in the first three blocks have a pooling size of 3 and strides of size 2. The first block has a max pooling layer, the second and third have an average pooling layer and the last block has a global average pooling layer.

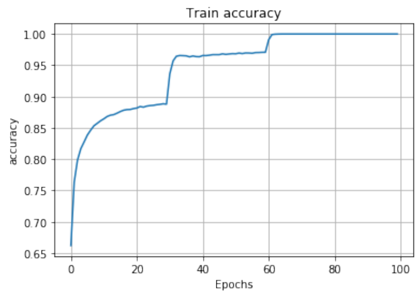
For each convolutional layer, we initialized the weighs using a normal distribution with 0 mean and standard deviation calculated by $\sqrt{2/(kernelsize * kernelsize * output)}$. Also, the biases were initialized as zeros.

We trained the network for 100 epochs. For each epoch, we trained all the rotated images from the training set. The batch size was set to 512, having 128x4 images. We used stochastic gradient descent algorithm as reducing loss function that is categorical cross entropy, with a momentum of 0.9. We also used a regularization of 5e-4 weight decay and a learning rate of 0.1. The learning rate was dropped by a factor of 5 after specific epoch numbers(30, 60, 80). The training data was randomly shuffled at each epoch.

After training our RotNet, we got 91.19% test accuracy.



[a]



[b]

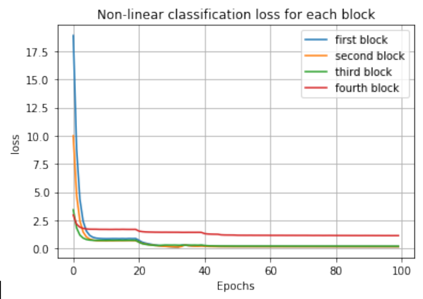
Figure 4: [a] The loss function of RotNet training, [b] The accuracy function of RotNet training

5.2 Best feature map

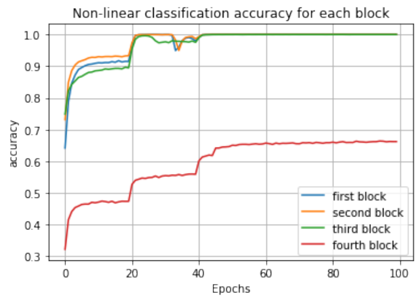
Once our RotNet model was trained, we tested how it performs for the non-linear classification depending on which of the feature maps of the different blocks we use. In order to do so, we took all the previously trained layers of the RotNet until the end of each block, froze them, and added three fully connected layers. The first two fully connected layers had 200 units, followed by batch normalization and ReLu activation function. The last fully connected layer had 10 units and a softmax activation function to classify the 10 labels. We used same parameters mentioned in Experiments. Finally, the learning rate was initialized to 0.1 and it was dropped by a factor of 5 in the 20^{th}, 40^{th}, 45^{th} and 50^{th} epochs.

For testing the performance of the fourth block, we had to change the global average pooling layer for a regular average pooling one in order to next stack the three fully connected layers.

The initialization of the weights samples were drawn from a truncated normal distribution centered on zero, with $\sqrt{2/outputsize}$.



[a]



[b]

Figure 5: [a]The loss function of non-linear classifier for each blocks [b] The accuracy function of non-linear classifier for each blocks

Table 3: The non-linear classification depending on the feature map of the different blocks. Test accuracy

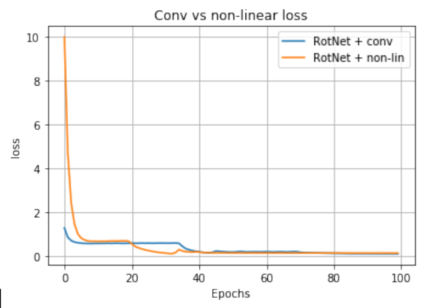
	Accuracy			
RotNet block	Block 1	Block 2	Block 3	Block 4
RotNet + non-linear	77.95%	**83.84%**	81.12 %	43.07%

From the plots and table above, we concluded that the second block is the one that has the best feature map to use for classification. We assume that the features of the convolutional blocks that follow the second one are more specific on the self-supervised task of rotation prediction, therefore their performance on the classification task is worse, specially the fourth block.

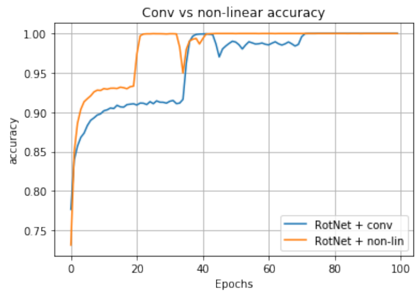
5.3 Non-linear vs convolutional

After selecting the second block as the best one to perform classification tasks, we wanted to test if a convolutional approach would be better than the previously mentioned non-linear one. For that, we defined a new architecture that consisted on stacking the first two blocks of the RotNet with the Class Conv Block defined in Table 2.

For the RotNet+conv training, the learning rate was initialized to 0.1 and it was dropped by a factor of 5 in the 35^{th}, 70^{th} and 85^{th} epochs.



[a]



[b]

Figure 6: [a]The loss function of RotNet+conv and RotNet + non-linear [b] The accuracy function of RotNet+conv and RotNet + non-linear

Table 4: Comparision between RotNet + non-linear classifier and RotNet + Conv. Test accuracy

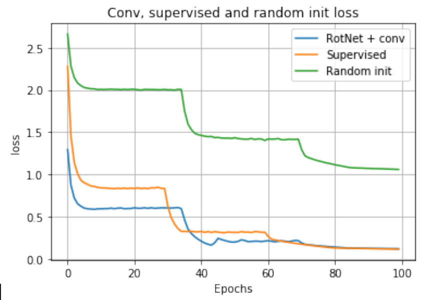
Model	Accuracy
RotNet + non-linear	83.84%
RotNet + Conv	**87.74%**

We can see from the table above that RotNet+conv has better accuracy when it comes to classification.

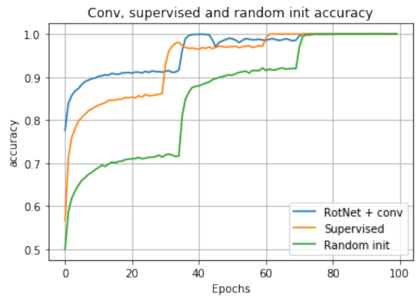
5.4 Convolutional, Random Initialization, Supervised learning

To have a better understanding on how useful is to use the blocks from the pre-trained RotNet model, we compared the performance of RotNet+conv with a supervised model and a random initialization of the tow RotNet blocks with a convolutional block.

All the tested networks have the same architecture, the only difference is that for RotNet+conv, the first two blocks are taken from RotNet and frozen while training. For the supervised model, the three convolutional blocks are randomly initialized and trained. The learning rate was defined like in RotNet. For the last model, the first two blocks were randomly initialized and frozen, and the third one was randomly initialized and trained. Also, the learning rate was defined like in RotNet+conv.



[a]



[b]

Figure 7: [a]The loss function of RotNet+conv, supervised and random init + conv [b] The accuracy function of RotNet+conv, supervised and random init + conv

Table 5: Results of RotNet+conv, supervised and Random Init+ conv. Test accuracy

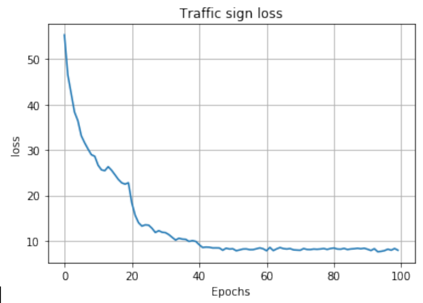
Model	Accuracy
Supervised NIN	89.72%
RotNet + Conv	**87.74%**
Random Init + Conv	10.01%

We can see that the performance of the RotNet+conv is better than the other two in the first epochs, which was the expected outcome. The supervised model is the one that gives the highest test accuracy. We can conclude that RotNet is a helpful tool to use in case of having few training data or little epochs for training, as the feature map extracts relevant information from the images that can be used for classification.

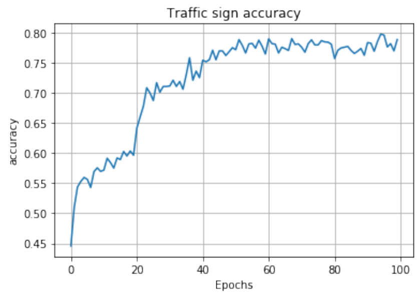
5.5 Traffic Sign

For this experiment, a lot of pre-processing was required, as the images of the dataset did not have the same dimension as the CIFAR-10 ones. The details of that pre-processing are explained in the Data section.

The network used for this regression task also uses the first two blocks of the RotNet model. These blocks are followed with a convolutional block that has the same labels as Block 3 defined in Table 2, but instead of having average pooling as the last layer, we put a Flatten layer. After that, we added two fully connected layers with 200 units, which had batch normalization and ReLu activation function. Also, the learning rate was initialized to 0.05 and it was dropped by a factor of 5 in the 20^{th}, 40^{th}, 45^{th} and 50^{th} epochs. The last layer is another fully connected layer of 4 units with no activation function. The loss function used in this case was mean squared error. To make sure that the RotNet blocks do extract relevant features we also tested the performance of the same architecture replacing the first two blocks by randomly initialized and frozen blocks.



[a]



[b]

Figure 8: [a]The loss function of traffic sing [b] The accuracy function of traffic sign

Table 6: The results of RotNet + Conv + non-linear and Random Init + Conv + non-linear in Traffic Sign dataset. Test accuracy

Method	Accuracy
RotNet + Conv + non-linear	69.91%
RandInit + Conv + non-linear	22.51%

The accuracy is not very high, as expected because RotNet was trained for classification purposes, but it still proves that RotNet does have a good feature map that extracts relevant information about the image that can be used for more than classification. In Figure 9 we can see some of the predictions the network made.

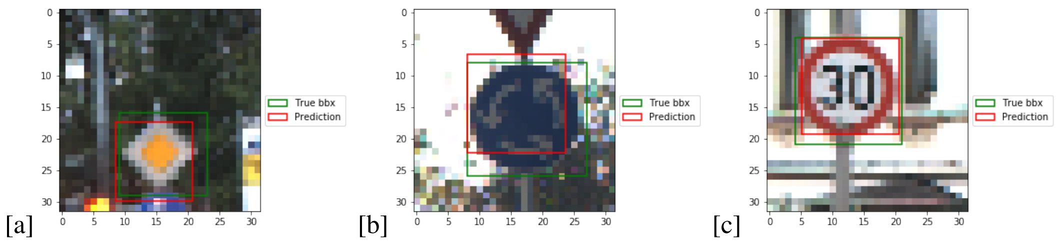


Figure 9: Some result images of bounding box on traffic sign

6 Conclusion

After the experiments, we can conclude that the second convolutional block of our RotNet provides a representative feature map for classification. Also, RotNet + Conv gives better results than the non-linear classifier. Although we got better test accuracy with the supervised learning, RotNet works better if we train for less epochs or have less data. Trained RotNet also works for regression problems but it is not as good as for classification tasks.

We couldn't get the same results as the paper that we followed but we came to the same conclusions that they have.

In order to test further the advantage of using RotNet we could have tested some other classification tasks with smaller datasets. It is supposed to perform better than a supervised approach, since with less data, during the training process the network would adapt to the training data set and would not generalize well. With the use of RotNet we could avoid that.

References

[1] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018.

[2] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[3] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[4] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, pages 1453–1460, 2011.