Very Deep Convolutional Networks And Random Forest Used In Image Processing Techniques

Vijay Swaroop A[1], Prekshitha S[2], Supriya P[3], Soujanya G Kiragi[4], Zaiba Farheen[5]
Atria Institute of Technology

Abstract- *Here, we inspect the outcome of the convolutional network depth on its correctness in the large-scale image recognition tasks. The chief role is thorough valuation of networks of increasing deepness by means of an architecture with small (3 × 3) convolution filters, and we also focus on one the algorithm called "Random Forest", which is a famous algorithm in Machine Learning ,This algorithm proves that it is efficient for the classification purpose. These outcomes in convolutional network were depended on our ImageNet Challenge 2014 submission. And in the next segments of the paper we get the understanding of the convolutional neural networks and the Random Forest also the differences between them, then finally we come to the conclusion of knowing the efficient algorithm in order to classify the images*

Keywords- Deep Convolutional Neural Networks, ImageNet, Feature extraction, Classification, Random Forest, Regression.

I. INTRODUCTION

Convolutional networks (ConvNets) have freshly enjoyed a big success in large-scale image and video recognition (Krizhevsky et al., 2012; Zeiler & Fergus, 2013; Sermanet et al., 2014; Simonyan & Zisserman, 2014) which is now possible by the large public image repositories, such as ImageNet (Deng et al., 2009), and high-performance computing systems, for imstance GPUs or large-scale distributed clusters (Dean et al., 2012).particularly,played a vital role in the progress of deep visual recognition architectures has been played by the ImageNet Large-Scale Visual Recog- nition Challenge (ILSVRC) (Russakovsky et al., 2014), which has served as a testbed for a few generations of large-scale image classification systems, from high-dimensional shallow feature en- codings (Perronnin et al., 2010) (the winner of ILSVRC-2011) to deep ConvNets (Krizhevsky et al., 2012) (the winner of ILSVRC-2012).
With ConvNets becoming more of a commodity in the computer vision field, a number of at- tempts have been made to improve the original architecture of Krizhevsky et al. (2012) in a bid to achieve better accuracy. For instance, the best-performing submissions to the ILSVRC- 2013 (Zeiler & Fergus, 2013; Sermanet et al., 2014) utilised smaller receptive window size and smaller stride of the first convolutional layer. Another line of improvements dealt with training and testing the networks densely over the whole image and over multiple scales (Sermanet et al., 2014; Howard, 2014). In this paper, we address another important aspect of ConvNet architecture design – its depth. To this end, we fix other parameters of the architecture, and steadily increase the depth of the network by adding more convolutional layers, which is feasible due to the use of very small (3 × 3) convolution filters in all layers.

As a result, we come up with significantly more accurate ConvNet architectures, which not only achieve the state-of-the-art accuracy on ILSVRC classification and localisation tasks, but are also applicable to other image recognition datasets, where they achieve excellent performance even when used as a part of a relatively simple pipelines (e.g. deep features classified by a linear SVM without fine-tuning). We have released our two best-performing models1 to facilitate further research.

The rest of the paper is organised as follows. In Sect. 2, we describe our ConvNet configurations. The details of the image classification training and evaluation are then presented in Sect. 3, and the configurations are compared on the ILSVRC classification task in Sect. 4. Sect. 5 concludes the paper. For completeness, we also describe and assess our ILSVRC-2014 object localisation system in Appendix A, and discuss the generalisation of very deep features to other datasets in Appendix B. Finally, Appendix C contains the list of major paper revisions.

II. CONVNET CONFIGURATIONS

To measure the improvement brought by the increased ConvNet depth in a fair setting, all our ConvNet layer configurations are designed using the same principles, inspired by Ciresan et al. (2011); Krizhevsky et al. (2012). In this section, we first describe a generic layout of our ConvNet configurations (Sect. 2.1) and then detail the specific configurations used in the evaluation (Sect. 2.2). Our design choices are then discussed and compared to the prior art in Sect. 2.3.

ARCHITECTURE

During training, the input to our ConvNets is a fixed-size 224 × 224 RGB image. The only pre- processing we do is subtracting the mean RGB value, computed on the training set, from each pixel. The image is passed through a stack of convolutional (conv.) layers, where we use filters with a very small receptive field: 3 × 3 (which is the smallest size to capture the notion of left/right, up/down,center). In one of the configurations we also utilise 1 × 1 convolution filters, which can be seen asa linear transformation of the input channels (followed by non-linearity). The convolution stride isfixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1 pixel for 3 × 3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2 × 2 pixel window, with stride 2.

A stack of convolutional layers (which has a different depth in different architectures) is followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 1000- way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks.

All hidden layers are equipped with the rectification (ReLU (Krizhevsky et al., 2012)) non-linearity. We note that none of our networks (except for one) contain Local Response Normalisation (LRN) normalisation (Krizhevsky et al., 2012): as will be shown in Sect. 4, such normalisation does not improve the performance on the ILSVRC dataset, but leads to increased memory con- sumption and computation time. Where applicable, the parameters for the LRN layer are those of (Krizhevsky et al., 2012).

CONFIGURATIONS

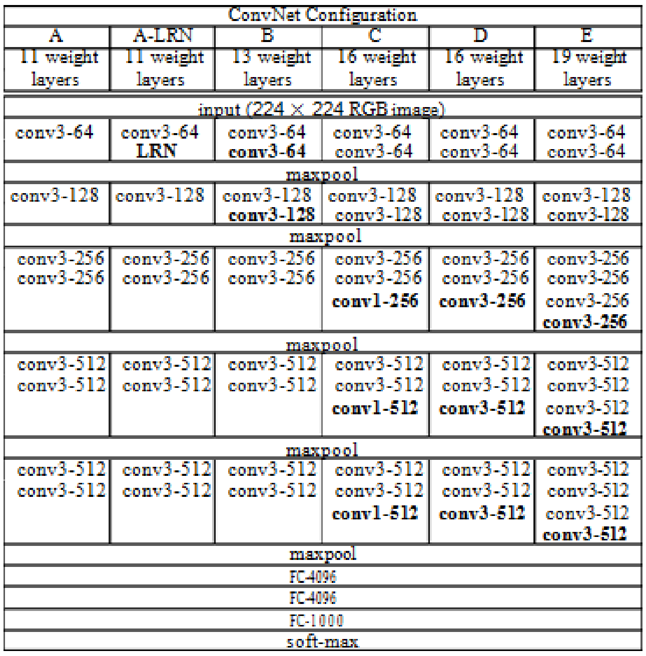
The ConvNet configurations, evaluated in this paper, are outlined in Table 1, one per column. In the following we will refer to the nets by their names (A–E). All configurations follow the generic design presented in Sect. 2.1, and differ only in the depth: from 11 weight layers in the network A (8 conv. and 3 FC layers) to 19 weight layers in the network E (16 conv. and 3 FC layers). The width of conv. layers (the number of channels) is rather small, starting from 64 in the first layer and then increasing by a factor of 2 after each max-pooling layer, until it reaches 512.

In Table 2 we report the number of parameters for each configuration. In spite of a large depth, the number of weights in our nets is not greater than the number of weights in a more shallow net with larger conv. layer widths and receptive fields (144M weights in (Sermanet et al., 2014)).

DISCUSSION

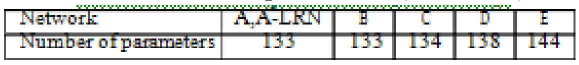
Our ConvNet configurations are quite different from the ones used in the top-performing entries of the ILSVRC-2012 (Krizhevsky et al., 2012) and ILSVRC-2013 competitions (Zeiler & Fergus, 2013; Sermanet et al., 2014). Rather than using relatively large receptive fields in the first conv. lay- ers (e.g. 11 × 11 with stride 4 in (Krizhevsky et al., 2012), or 7 × 7 with stride 2 in (Zeiler & Fergus, 2013; Sermanet et al., 2014)), we use very small 3 × 3 receptive fields throughout the whole net, which are convolved with the input at every pixel (with stride 1). It is easy to see that a stack of two 3 × 3 conv. layers (without spatial pooling in between) has an effective receptive field of 5 × 5; three

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv(receptive field size)-(number of channels)". The ReLU activation function is not shown for brevity.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).



Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

such layers have a 7 × 7 effective receptive field. So what have we gained by using, for instance, a stack of three 3 × 3 conv. layers instead of a single 7 × 7 layer? First, we incorporate three non-linear rectification layers instead of a single one, which makes the decision function more discriminative.

Second, we decrease the number of parameters: assuming that both the input and the output of a three-layer 3 × 3 convolution stack has C channels, the stack is parametrised by $3\ .3^2C^2\Sigma = 27C^2$

weights; at the same time, a single 7 × 7 conv. layer would require $7^2C^2 = 49C^2$ parameters, i.e. 81% more. This can be seen as imposing a regularisation on the 7 × 7 conv. filters, forcing them to have a decomposition through the 3 × 3 filters (with non-linearity injected in between).

The incorporation of 1 × 1 conv. layers (configuration C, Table 1) is a way to increase the non-linearity of the decision function without affecting the receptive fields of the conv. layers. Even though in our case the 1 × 1 convolution is essentially a linear projection onto the space of the same dimensionality (the number of input and output channels is the same), an additional non-linearity is introduced by the rectification function. It should be noted that 1 × 1 conv. layers have recently been utilised in the "Network in Network" architecture of Lin et al. (2014).

Small-size convolution filters have been previously used by Ciresan et al. (2011), but their nets are significantly less deep than ours, and they did not evaluate on the large-scale ILSVRC dataset. Goodfellow et al. (2014) applied deep ConvNets (11 weight layers) to the task of street number recognition, and showed that the increased depth led to better performance. GoogLeNet (Szegedy et al., 2014), a top-performing entry of the ILSVRC-2014 classification task, was developed independently of our work, but is similar in that it is based on very deep ConvNets (22 weight layers) and small convolution filters (apart from 3 × 3, they also use 1 × 1 and 5 × 5convolutions). Their network topology is, however, more complex than ours, and the spatial reso-lution of the feature maps is reduced more aggressively in the first layers to decrease the amount of computation. As will be shown in Sect. 4.5, our model is outperforming that of Szegedy et al. (2014) in terms of the single-network classification accuracy.

III. CLASSIFICATION FRAMEWORK

In the previous section we presented the details of our network configurations. In this section, we describe the details of classification ConvNet training and evaluation.

TRAINING

The ConvNet training procedure generally follows Krizhevsky et al. (2012) (except for sampling the input crops from multi-scale training images, as explained later). Namely, the training is carried out by optimising the multinomial logistic regression objective using mini-batch gradient descent (based on back-propagation (LeCun et al., 1989)) with momentum. The batch size was set to 256, momentum to 0.9. The training was regularised by weight decay (the L_2 penalty multiplier set to$5 \cdot 10^{-4}$) and dropout regularisation for the first two fully-connected layers (dropout ratio set to 0.5). The learning rate was initially set to 10^{-2}, and then decreased by a factor of 10 when the validation set accuracy stopped improving. In total, the learning rate was decreased 3 times, and the learning was stopped after 370K iterations (74 epochs). We conjecture that in spite of the larger number of parameters and the greater depth of our nets compared to (Krizhevsky et al., 2012), the nets required less epochs to converge due to (a) implicit regularisation imposed by greater depth and smaller conv. filter sizes; (b) pre-initialisation of certain layers.

The initialisation of the network weights is important, since bad initialisation can stall learning due to the instability of gradient in deep nets. To circumvent this problem, we began with training the configuration A (Table 1), shallow enough to be trained with random initialisation. Then, when training deeper architectures, we initialised the first four convolutional layers and the last three fully- connected layers with the layers of net A (the intermediate layers were initialised randomly). We did not decrease the learning rate for the pre-initialised layers, allowing them to change during learning.

For random initialisation (where applicable), we sampled the weights from a normal distribution with the zero mean and 10^{-2} variance. The biases were initialised with zero. It is worth noting that after the paper submission we found that it is possible to initialise the weights without pre-training by using the random initialisation procedure of Glorot & Bengio (2010).

To obtain the fixed-size 224×224 ConvNet input images, they were randomly cropped from rescaled training images (one crop per image per SGD iteration). To further augment the training set, the crops underwent random horizontal flipping and random RGB colour shift (Krizhevsky et al., 2012).

Training image rescaling is explained below.

Training image size. Let S be the smallest side of an isotropically-rescaled training image, from which the ConvNet input is cropped (we also refer to S as the training scale). While the crop size is fixed to 224 × 224, in principle S can take on any value not less than 224: for S = 224 the crop will capture whole-image statistics, completely spanning the smallest side of a training image; forS $\gg$ 224 the crop will correspond to a small part of the image, containing a small object or an object part.

We consider two approaches for setting the training scale S. The first is to fix S, which corresponds to single-scale training (note that image content within the sampled crops can still represent multi- scale image statistics). In our experiments, we evaluated models trained at two fixed scales: S = 256 (which has been widely used in the prior art (Krizhevsky et al., 2012; Zeiler & Fergus, 2013; Sermanet et al., 2014)) and S = 384. Given a ConvNet configuration, we first trained the network using S = 256. To speed-up training of the S = 384 network, it was initialised with the weights pre-trained with S = 256, and we used a smaller initial learning rate of 10^{-3}.

The second approach to setting S is multi-scale training, where each training image is individually rescaled by randomly sampling S from a certain range [S_{min}, S_{max}] (we used S_{min} = 256 and S_{max} = 512). Since objects in images can be of different size, it is beneficial to take this into account during training. This can also be seen as training set augmentation by scale jittering, where a single model is trained to recognise objects over a wide range of scales. For speed reasons, we trained multi-scale models by fine-tuning all layers of a single-scale model with the same configuration, pre-trained with fixed S = 384.

TESTING

At test time, given a trained ConvNet and an input image, it is classified in the following way. First, it is isotropically rescaled to a pre-defined smallest image side, denoted as Q (we also refer to it as the test scale). We note that Q is not necessarily equal to the training scale S (as we will show in Sect. 4, using several values of Q for each S leads to improved performance). Then, the networkis applied densely over the rescaled test image in a way similar to (Sermanet et al., 2014). Namely, the fully-connected layers are first converted to convolutional layers (the first FC layer to a 7 × 7 conv. layer, the last two FC layers to 1 × 1 conv. layers). The resulting fully-convolutional net is then applied to the whole (uncropped) image. The result is a class score map with the number ofchannels equal to the number of classes, and a variable spatial resolution, dependent on the input image size. Finally, to obtain a fixed-size vector of class scores for the image, the class score map is spatially averaged (sum-pooled). We also augment the test set by horizontal flipping of the images; the soft-max class posteriors of the original and flipped images are averaged to obtain the final scores for the image.

Since the fully-convolutional network is applied over the whole image, there is no need to sample multiple crops at test time (Krizhevsky et al., 2012), which is less efficient as it requires network re-computation for each crop. At the same time, using a large set of crops, as done by Szegedy et al. (2014), can lead to improved accuracy, as it results in a finer sampling of the input image compared to the fully-convolutional net. Also, multi-crop evaluation is complementary to dense evaluation due to different convolution boundary conditions: when applying a ConvNet to a crop, the convolved feature maps are padded with zeros, while in the case of dense evaluation the padding for the same crop naturally comes from the neighbouring parts of an image (due to both the convolutions and spatial pooling), which substantially increases the overall network receptive field, so more context is captured. While we believe that in practice the increased computation time of multiple crops doesnot justify the potential gains in accuracy, for reference we also evaluate our networks using 50 crops per scale (5 × 5 regular grid with 2 flips), for a total of 150 crops over 3 scales, which is comparable to 144 crops over 4 scales used by Szegedy et al. (2014).

IMPLEMENTATION DETAILS

Our implementation is derived from the publicly available C++ Caffe toolbox (Jia, 2013) (branched out in December 2013), but contains a number of significant modifications, allowing us to perform training and evaluation on multiple GPUs installed in a single system, as well as train and evaluate on full-size (uncropped) images at multiple scales (as described above). Multi-GPU training exploits data parallelism, and is carried out by splitting each batch of training images into several GPU batches, processed in parallel on each GPU. After the GPU batch gradients are computed, they are averaged to obtain the gradient of the full batch. Gradient computation is synchronous across the GPUs, so the result is exactly the same as when training on a single GPU.

While more sophisticated methods of speeding up ConvNet training have been recently pro- posed (Krizhevsky, 2014), which employ model and data parallelism for different layers of the net, we have found that our conceptually much simpler scheme already provides a speedup of 3.75 times on an off-the-shelf 4-GPU system, as compared to using a single GPU. On a system equipped with four NVIDIA Titan Black

GPUs, training a single net took 2–3 weeks depending on the architecture.

IV. RANDOM FOREST

Random Forest is the most popular and more powerful supervised machine learning algorithm.It is capable of performing both regression and classification tasks.As the name suggests,this algorithm creates a random forest with number of decision trees.More the trees in the forest, more Robust the prediction,Thus it gives up the high accuracy

WORKING OF RANDOM FOREST

To classify a new object based on the attribute.Each tree gives a classification and we say the tree votes for that class.

We first choose the classification having more votes of all the other tree in the forest

And in the regression takes the averege of the output by different trees.

ADVANTAGES

- Some Random forest algorithm can be used for classification and regression tasks.
- Handle the missing values and maintains accuracy for missing data.
- When we have more trees in the forest ,it wont overfit the model.
- It has the power to handle large dataset with higher dimensionality.

DISADVANTAGES

- Good job at classification but was as good as for regression.
- We have very little control on what the model does.

APPLICATIONS

- It can be used in the banking sectors.
- It can be used in the meddicine sectors to identify the correct combination of components to validate the medicine.
- It is also used to identify disease by analyzing the patient's medical reccord.
- In computer vision Random Forest is used for image classification.

PSEUDOCODE

- Assume number of cases in the training set is 'N'.Then sample of these 'N' cases is taken at random but with replacement
- If there are 'M' input variables or features,a number m<M is specified such that at each node m variables are selected at random out of the 'M'.The best split on these m is used to split the node.The value of 'm' is held constant while we grow the forest
- Each tree is grown to the largest extent possible and there is no pruning.
- Predict new data by aggregating the predictions of the trees(i.e majority votes for classification average for regression).

IMPLEMENTATION

- Assume we found a 1000 Random decision trees.
- We need to pass the test features through the Rows of each randomly created trees
- Say if we have 1000 Random decision trees to create the forest.
- If an image contain hand,each Random forest will predict the different outcome or class for the same test features.
- Let us consider the random set of features for example a finger
- Suppose 100 Random decision trees predict some unit target,such as a finger,thumb.
- Then the votes for finger is given out of 100
- If finger is getting the highest votes then the final random forest returns the finger to predict the target,this concept of voting is known as the "majority voting".
- It also predicts rest of the fingers to be the fingers then the high level decision tree can vote that the image is a hand

This is why Random Forest is known as "Ensemble macline learning algorithm", where ensembles are divide and conquer approach .

V. CONCLUSION

In this work we evaluated very deep convolutional networks (up to 19 weight layers) for large- scale image classification. It was demonstrated that the representation depth is beneficial for the classification accuracy, and that state-of-the-art performance on the ImageNet challenge dataset can be achieved using a conventional ConvNet architecture (LeCun et al., 1989; Krizhevsky et al., 2012) with substantially increased depth. In the appendix, we also show that our models generalise well to a wide range of tasks and datasets, matching or outperforming more complex recognition pipelines built around less deep image representations. Our results yet again confirm the importance of depth in visual representations.Deep Convolutional Neural Networks are very efficient in feature extraction for large-scale unstructured data,but Random forest is used for the structured data classification as it proves that it is efficient in classifying the small-scale structured data.

VI. ACKNOWLEDGEMENTS

This work was supported by ERC grant VisRec no. 228180. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used for this research.

REFERENCES

[1] Bell, S., Upchurch, P., Snavely, N., and Bala, K. Material recognition in the wild with the materials in context database. *CoRR*, abs/1412.0623, 2014.

[2] Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. Return of the devil in the details: Delving deep into convolutional nets. In *Proc. BMVC.*, 2014.

[3] Cimpoi, M., Maji, S., and Vedaldi, A. Deep convolutional filter banks for texture recognition and segmentation. *CoRR*, abs/1411.6836, 2014.

[4] Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. Flexible, high performance convolutional neural networks for image classification. In *IJCAI*, pp. 1237–1242, 2011.

[5] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., and Ng, A. Y. Large scale distributed deep networks. In *NIPS*, pp. 1232–1240, 2012.

[6] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009.

[7] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013.

[8] Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C., Winn, J., and Zisserman, A. The Pascal visual object classes challenge: A retrospective. *IJCV*, 111(1):98–136, 2015.

[9] Fei-Fei, L., Fergus, R., and Perona, P. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *IEEE CVPR Workshop of Generative Model Based Vision*, 2004.

[10] Girshick, R. B., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524v5, 2014. Published in Proc. CVPR, 2014.

[11] Gkioxari, G., Girshick, R., and Malik, J. Actions and attributes from wholes and parts. *CoRR*, abs/1412.2604, 2014.

[12] Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS*, volume 9, pp. 249–256, 2010.

[13] Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. Multi-digit number recognition from street view imagery using deep convolutional neural networks. In *Proc. ICLR*, 2014.

[14] Griffin, G., Holub, A., and Perona, P. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

[15] He, K., Zhang, X., Ren, S., and Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729v2, 2014.

[16] Hoai, M. Regularized max pooling for image categorization. In *Proc. BMVC.*, 2014.

[17] Howard, A. G. Some improvements on deep convolutional neural network based image classification. In *Proc. ICLR*, 2014.

[18] Jia, Y. Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/, 2013.

[19] Karpathy, A. and Fei-Fei, L. Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306, 2014.

[20] Kiros, R., Salakhutdinov, R., and Zemel, R. S. Unifying visual-semantic embeddings with multimodal neural language models. *CoRR*, abs/1411.2539, 2014.

[21] Krizhevsky, A. One weird trick for parallelizing convolutional neural networks. *CoRR*, abs/1404.5997, 2014. Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural net- works. In *NIPS*, pp. 1106–1114, 2012.

[22] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropa- gation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[23] Lin, M., Chen, Q., and Yan, S. Network in network. In *Proc. ICLR*, 2014.

[24] Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[25] Oquab, M., Bottou, L., Laptev, I., and Sivic, J. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. In *Proc. CVPR*, 2014.
[26] Perronnin, F., Sa´nchez, J., and Mensink, T. Improving the Fisher kernel for large-scale image classification. In *Proc. ECCV*, 2010.
[27] Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. CNN Features off-the-shelf: an Astounding Baseline for Recognition. *CoRR*, abs/1403.6382, 2014.
[28] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.
[29] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In *Proc. ICLR*, 2014.
[30] Simonyan, K. and Zisserman, A. Two-stream convolutional networks for action recognition in videos. *CoRR*, abs/1406.2199, 2014. Published in Proc. NIPS, 2014.
[31] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich,Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.
[32] Wei, Y., Xia, W., Huang, J., Ni, B., Dong, J., Zhao, Y., and Yan, S. CNN: Single-label to multi-label. *CoRR*, abs/1406.5726, 2014.
[33] Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. Published in Proc. ECCV, 2014.