An Analysis of Unsupervised Pre-training in Light of Recent Advances

Tom Le Paine*, Pooya Khorrami*, Wei Han, Thomas S. Huang
Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
Urbana, IL 61801
`paine1,pkhorra2,weihan3,t-huang1@illinois.edu`

Abstract

Convolutional neural networks perform well on object recognition because of a number of recent advances: rectified linear units (ReLUs), data augmentation, dropout, and large labelled datasets. Unsupervised data has been proposed as another way to improve performance. Unfortunately, unsupervised pre-training is not used by state-of-the-art methods leading to the following question: Is unsupervised pre-training still useful given recent advances? If so, when? We answer this in three parts: we 1) develop an unsupervised method that incorporates ReLUs and recent unsupervised regularization techniques, 2) analyze the benefits of unsupervised pre-training compared to data augmentation and dropout on CIFAR-10 while varying the ratio of unsupervised to supervised samples, 3) verify our findings on STL-10. We discover unsupervised pre-training, as expected, helps when the ratio of unsupervised to supervised samples is high, and surprisingly, hurts when the ratio is low. We also use unsupervised pre-training with additional color augmentation to achieve near state-of-the-art performance on STL-10.

1 Introduction

We analyze the benefits of unsupervised pre-training in the context of recent deep learning innovations including: rectified linear units, data augmentation, and dropout. Recent work shows that convolutional neural networks (CNNs) can achieve state-of-the-art performance for object classification (Krizhevsky et al. (2012)) and object detection (Girshick et al. (2013)), when there is enough training data. However, in many cases there is a dearth of labeled data. In these cases regularization is necessary for good results. The most common types of regularization are data augmentations (Krizhevsky et al. (2012); Dosovitskiy et al. (2014)) and dropout (Hinton et al. (2012)). Another form of regularization, unsupervised pre-training (Hinton et al. (2006); Bengio et al. (2007); Erhan et al. (2010)), has recently fallen out of favor.

While there has been significant work in unsupervised learning, most of these works came before rectified linear units, which significantly help training deep supervised neural networks, and before simpler regularization schemes for unsupervised learning, such as zero-bias with linear encoding for auto-encoders (Memisevic et al. (2014)).

We train an unsupervised method that takes advantage of these improvements we call Zero-bias Convolutional Auto-encoders (CAEs). Previous work showed that pre-trained tanh CAEs achieved an increase in performance over randomly initialized tanh CNNs. We conduct this experiment with our zero-bias CAE and observe a larger boost in performance.

We analyze the effectiveness of our technique when combined with the popular regularization techniques used during supervised training on CIFAR-10 while varying the ratio of unsupervised to supervised samples. We do this comparing against randomly initialized CNNs without any additional regularization. We find that, when ratio is large, unsupervised pre-training provides useful regularization, increasing test set performance. When the ratio is small, we find that unsupervised pre-training hurts performance.

---

*- Authors contributed equally to this work.

We verify our finding that unsupervised pre-training can boost performance when the ratio of unsupervised to supervised samples is high by running our algorithm on the STL-10 dataset, which has a ratio of 100:1. As expected, we observe an improvement (3.87%). When combined with additional color augmentation, we achieve near state-of-the-art results. Our unsupervised regularization still yields an improvement of (1.69%).

We will begin by reviewing related work on fully-connected and convolutional auto-encoders. In Section 3, we will present our method and how it is trained both during unsupervised pre-training and supervised fine-tuning. We present our results on the CIFAR-10 and STL-10 datasets in Section 4, and in Section 5 we conclude the paper.

2   RELATED WORK

Many methods have used unsupervised learning to learn parameters, which are subsequently used to initialize a neural network to be trained on supervised data. These are called unsupervised pre-training, and supervised fine-tuning respectively. We will highlight some of the unsupervised learning methods related to our work.

2.1   AUTO-ENCODERS

One of the most widely-used models for unsupervised learning, an auto-encoder is a model that learns a function that minimizes the squared error between the input $x \in \mathbb{R}^n$ and its reconstruction $r(x)$:

$$L = \|x - r(x)\|_2^2 \tag{1}$$

$$r(x) = W_d^T f(W_e x + b) + c \tag{2}$$

In the above equation, W_e represents the weight matrix that transforms the input, x into some hidden representation, b is vector of biases for each hidden unit and $f(\cdot)$ is some nonlinear function. Commonly chosen examples for $f(\cdot)$ include the sigmoid and hyperbolic tangent functions. Meanwhile, W_d is the weight matrix that maps back from the hidden representation to the input space and c is a vector of biases for each input (visible) unit. These parameters are commonly learned by minimizing the loss function over the training data via stochastic gradient descent.

When no other constraints are imposed on the loss function, the auto-encoder weights tend to learn the identity function. To combat this, some form of regularization must imposed upon the model so that the model can uncover the underlying structure in the data. Some forms of regularization include adding noise to the input units (Vincent et al. (2010)) and requiring the hidden unit activations be sparse (Coates et al. (2011)) or have small derivatives (Rifai et al. (2011)). These models are known as de-noising, sparse, and contractive auto-encoders respectively. A more recent work by Memisevic et al. (2014) showed that training an auto-encoder with rectified linear units (ReLU) caused the activations to form tight clusters due to having negative bias values. They showed that using thresholded linear (TLin) or thresholded rectifier (TRec) activations with no bias can allow one to train an auto-encoder without the need for additional regularization.

2.2   CONVOLUTIONAL AUTO-ENCODERS

While the aforementioned fully-connected techniques have shown impressive results, they do not directly address the structure of images. Convolutional neural networks (CNNs) (LeCun et al. (1998); Lee et al. (2009)) present a way to reduce the number of connections by having each hidden unit only be responsible for a small local neighborhood of visible units. Such schemes allow for dense feature extraction followed by pooling layers which when stacked could allow the network to learn over larger and larger receptive fields. Convolutional auto-encoders (CAEs) combined aspects from both auto-encoders and convolutional neural nets making it possible to extract highly localized patch-based information in an unsupervised fashion. There have been several works in this area including Jarrett et al. (2009) and Zeiler et al. (2010). Both rely on sparse coding to force their unsupervised learning to learn non-trival solutions. Zeiler et al. (2011) extended this work by introducing pooling/unpooling and visualizing how individual feature maps at different layers influenced specific portions of the reconstruction. These sparse coding approaches had limitations because they used

an iterative procedure for inference. A later work by Masci et al. (2011) trained deep feed forward convolutional auto-encoders, using only max-pooling and saturating tanh non-linearities as a form of regularization, while still showing a modest improvement over randomly initialized CNNs. While tanh was a natural choice at the time, Krizhevsky et al. (2012) showed that ReLUs are more suitable for learning given their non-saturating behavior.

3 OUR APPROACH

Our method's training framework can be broken up into two phases: (i) unsupervised pre-training and (ii) supervised fine-tuning. We describe those in more detail below.

3.1 UNSUPERVISED PRE-TRAINING

Our method incorporates aspects of previous unsupervised learning methods in order to learn salient features, yet be efficient to train. Our model is similar to the deconvolutional network in Zeiler et al. (2011) where the cost we minimize at each layer is the mean square error on the original image. However, unlike the network in Zeiler et al. (2011), our method does not use any form of sparse coding. Our model also is similar to that of Masci et al. (2011), however we improve upon it by introducing regularization in the convolutional layers through the use of zero-biases and ReLUs as discussed in Memisevic et al. (2014).

We now describe the model architecture in detail. Like the previous work described above, our model involves several encoding modules followed by several decoding modules. A single encoding module $E_l(\cdot)$ consists of a convolution layer F_l, a nonlinearity $f(\cdot)$, followed by a pooling layer P_{s_l} with switches s_l.

$$E_l(x) = P_{s_l} f(F_l x) \tag{3}$$

Each encoding module has an associated decoding module D_l, which unpools using E_l pooling switches s_l and deconvolves with E_l's filters, (i.e. F_l^T).

$$D_l(x) = F_l^T U_{s_l} x \tag{4}$$

A two layer network can be written as:

$$r(x) = D_1(D_2(E_2(E_1(x)))) \tag{5}$$

We train each encoder/decoder pair in a greedy fashion (i.e. first a 1 layer CAE, then a 2 layer CAE, etc.) while keeping the parameters of previous layers fixed. Like Zeiler et al. (2011), we compute the cost by taking the mean squared error between the original image and the network's reconstruction of the input. Thus, the costs for a one layer network ($C_1(x)$) and two layer network ($C_2(x)$) would be expressed in the following manner:

$$C_1(x) = \|x - D_1(E_1(x))\|_2^2 \tag{6}$$
$$C_2(x) = \|x - D_1(D_2(E_2(E_1(x))))\|_2^2 \tag{7}$$

We regularize our learned representation by fixing the biases of our convolutional and deconvolutional layers at zero and using ReLUs as our activation function during encoding. We use linear activations for our decoders. Unlike the work by Memisevic et al. (2014) which analyzes fully-connected auto-encoders, our work is the first, to our knowledge, that trains zero-bias CAEs for unsupervised learning.

3.1.1 UNSUPERVISED WEIGHT INITIALIZATION

Weight initialization is often a key component of successful neural network training. For ReLU's it is important to ensure the input to the ReLU is greater than 0. This can be achieved by setting the bias appropriately. This cannot be done for zero-bias auto-encoders. Instead we use two methods

for initializing the weights to achieve this 1) in the first layer, we initialize each of the filters to be a randomly drawn patch from the dataset, 2) on the later layers, we sample weights from a Gaussian distribution and find the nearest orthogonal matrix by taking the singular value decomposition (SVD) of the weight matrix and setting all of the singular values to one. For CNNs we must take into account the additive effect of overlapping patches thus we weight each filter by a 2D hamming window to prevent intensity build-up.

3.2 Supervised Fine-tuning

After the weights of the CAE have been trained, we remove all of the decoder modules and leave just the encoding modules. We add an additional fully-connected layer and a softmax layer to the pre-trained encoding modules. The weights of these layers are drawn from a Gaussian distribution with zero mean and standard deviation of $k/\sqrt{N_{FAN\_IN}}$, where k is drawn uniformly from $[0.2, 1.2]$.

3.3 Training

For both unsupervised and supervised training we use stochastic gradient descent with a constant momentum of 0.9, and a weight decay parameter of 1e-5. We select the highest learning rate that doesn't explode for the duration of training. For these experiments we do not anneal the learning rate. The only pre-processing we do to each patch is centering (i.e. mean subtraction) and scaling to unit variance.

4 Experiments and Analysis

4.1 Datasets

We run experiments on two natural image datasets, CIFAR-10 (Krizhevsky and Hinton (2009)) and STL-10 (Coates et al. (2011)). CIFAR-10 is a common benchmark for object recognition. Many unsupervised and supervised neural network approaches have been tested on it. It consists of 32x32 pixel color images drawn from 10 object categories. It has 50,000 training images, and 10,000 testing images. STL-10 is also an object recognition benchmark, but was designed to test unsupervised learning algorithms, so it has a relatively small labeled training set of 500 images per class, and an additional unsupervised set which contains 100,000 unlabeled images. The test set contains 800 labeled images per class. All examples are 96x96 pixel color images.

4.2 CIFAR-10

On CIFAR-10, we train a network with structure similar to Masci et al. (2011), so that we can directly show the benefits of our modifications. The network consists of three convolutional layers with 96, 144, and 192 filters respectively. The filters in the first two layers are of size 5x5 while the filters in the third layer are of size 3x3. We also add 2x2 max pooling layers after the first two convolutional layers. There is also a full-connected layer with 300 hidden units followed by a softmax layer with 10 output units. All of our nets were trained using our own open source neural network library [1].

As stated in the methods section, we first train our unsupervised model on 100% of the training images, do supervised fine-tuning, and report overall accuracy on the test set. We 1) present qualitative results of unsupervised learning, 2) show our zero-bias convolutional auto-encoder performs well compared to previous convolutional auto-encoder work by Masci et al. (2011) developed before the popularization of rectified linear units, and zero-bias auto-encoders, 3) we show our analysis of various regularization techniques, and vary the ratio of unsupervised to supervised data, 4) for completeness we report our best results when training on the full CIFAR-10 dataset, however this is not the main point of this work.

4.2.1 Qualitative Results

One way in which we ensure the quality of our learned representation is by inspecting the first layer filters. We visualize the filters learned by our model in Figure 1. So that we can directly compare

---

[1]https://github.com/ifp-uiuc/anna

with the filters presented in Masci et al. (2011), we trained an additional zero-bias convolutional auto-encoder with filters of size 7x7x3 (instead of 5x5x3) in the first layer. From Figure 1, we can see that, indeed, our model is able to capture interpretable patterns such as Gabor-like oriented edges (both color and intensity) and center-surrounds.

4.2.2 Unsupervised Pre-training for Tanh CAEs and Zero-bias CAEs

For our quantitative experiments, we first compare the performance of the tanh CAE proposed by Masci et al. (2011) with our zero-bias CAE. In their paper, Masci et al. (2011) trained a tanh CNN from a random initialization and compared it with one pre-trained using a tanh CAE. They also added 5% translations as a form of data augmentation. We re-conduct this experiment using a zero-bias CNN trained from a random initialization, and compare it to one pre-trained using our zero-bias CAE.
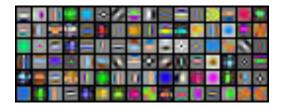

Figure 1: First layer filters learned by our zero-bias convolutional auto-encoder. Each filter has dimension 7x7x3. (Best viewed in color.) For direct comparison with tanh CAE please see Masci et al. (2011) Figure 2c.

In Table 1 we compare the improvements of our model with that of Masci et al. (2011)'s, on various subsets of CIFAR-10. As expected, the zero-bias CNN (a ReLU CNN without bias parameters) performs significantly better than the tanh CNN (2.53%, 8.53%, 5.23%). More interestingly, notice that on each subset, compared to Masci et al. (2011) our pre-trained model shows similar or better performance over the randomly initialized CNN. When the ratio of unsupervised to supervised data is high, we experience an 8.44% increase in accuracy as opposed to Masci et al. (2011)'s 3.22% increase.

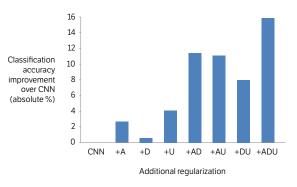### 4.2.3 Analysis of regularization methods

Next, we analyze how different supervised regularization techniques affect our model's performance. Specifically, we consider the effects of dropout, data augmentation (via translations and horizontal flips), unsupervised pre-training (with our zero-bias CAE) and their combinations. We compare each regularization technique to a zero-bias CNN trained from random initialization without any regularization (labeled CNN in Figure 2). Figure 2 shows the classification accuracy improvement over CNN for each type of regularization both individually and together.
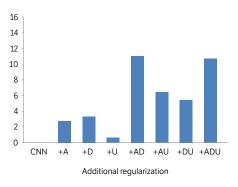
We perform this analysis for subsets of CIFAR-10 with different unsupervised to supervised sample ratios ranging from 50:1 to 1:1, by fixing the unsupervised data size, and varying the number of supervised examples. It is important to note that as this ratio approaches 1:1, the experimental setup favors data augmentation and dropout because the number of virtual supervised samples is larger than number of unsupervised samples.

In Figure 2a, where the ratio of unsupervised to supervised samples is 50:1, there are three notable effects: (i) unsupervised pre-training alone yields a larger improvement (4.09%) than data augmentation (2.67%) or dropout (0.59%), (ii) when unsupervised pre-training is combined with either data augmentation or dropout, the improvement is greater than the sum of the individual contributions, (iii) we experience the largest gains (15.86%) when we combine all three forms of regularization.

Table 1: Comparison between Tanh CAE (Masci et al. (2011)) and our model on various subsets of CIFAR-10.
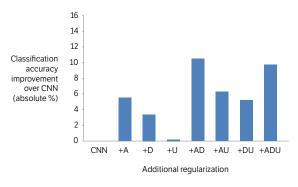
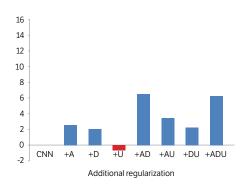| Unsupervised to supervised ratio (Samples per Class)	50:1 (100)	10:1 (500)	5:1 (1000)	1:1 (5000)
Tanh CNN - Masci et al. (2011)	44.48 %	—	64.77 %	77.50 %
Tanh CAE - Masci et al. (2011)	47.70 %	—	65.65 %	78.20 %
Zero-bias CNN	47.01 %	64.76 %	73.30 %	82.73 %
Zero-bias CAE	**55.45 %**	**68.42 %**	**74.06 %**	**83.64 %**

(a) 50:1 unsupervised to supervised sample ratio (100 samples per class), baseline CNN: 44.3%

(b) 10:1 unsupervised to supervised sample ratio (500 samples per class), baseline CNN: 62.0%

(c) 5:1 unsupervised to supervised sample ratio (1000 samples per class), baseline CNN: 67.8%

(d) 1:1 unsupervised to supervised sample ratio (5000 samples per class), baseline CNN: 80.2%

Figure 2: Analysis of the effects of different types of regularization (A: data augmentation, D: dropout, U: unsupervised learning), individually and jointly, on different subsets of CIFAR-10.

We see that effect (ii) is also observed in the case where the ratio of unsupervised to supervised samples is 10:1 (Figure 2b), and to a lesser extent when the ratio is 5:1 (Figure 2c). Unfortunately, effects (i) and (iii) are not observed when the ratio of unsupervised to supervised samples decreases. We will elaborate on effect (i) below.

In Figure 3, we observe that the improvement in performance from unsupervised learning decreases rapidly as the ratio of unsupervised to supervised samples decreases. Surprisingly, when the ratio is 1:1, we see that unsupervised learning actually hurts performance (-0.67%).

4.2.4 COMPARISON WITH EXISTING METHODS

We also compare the performance of our algorithm on the full CIFAR-10 dataset with other techniques in Table 2, though we show above our method performs worse when the ratio of unsupervised to supervised samples is 1:1. We outperform all methods that use unsupervised pre-training (Masci et al. (2011), Coates et al. (2011), Dosovitskiy et al. (2014), Lin and Kung (2014)), however we are not competitive with supervised state-of-the-art. We include some representative supervised methods in Table 2.
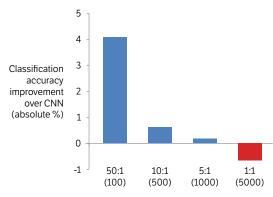
4.3 STL-10

Next, we assess the effects of unsupervised pre-training on STL-10. From the CIFAR-10 experiments, it is clear unsupervised pre-training can be beneficial if the unsupervised dataset is much larger than the supervised dataset. STL-10 was designed with this in mind, and has a ratio of unsupervised to supervised data of 100:1. So we experimentally show this benefit.

We design our network to have structure similar to Dosovitskiy et al. (2014), to ease comparison. The network used consists 3 convolutional layers with 64, 128, and 256 filters in each layer, a fully-connected layer with 512 units, and a softmax layer with 10 output units. We also apply max-pooling layers of size 2x2 after the first two convolutional layers and quadrant pooling after the third convolutional layer.

We train the zero-bias CAE on 100,000 unlabeled images. We then fine-tune the network on each of the 10 provided splits of training set, each consisting of 1000 samples (100 samples per class), and evaluate all of them on the test set. The accuracies are subsequently averaged to obtain the final recognition accuracy. Similar to our CIFAR-10 experiments, we also train a zero-bias CNN with the same structure as our zero-bias CAE on each of the splits to further highlight the benefits of unsupervised learning.

Table 3 presents our results on the STL-10 dataset and compares them with other methods. As expected, unsupervised pre-training gives a 3.87% increase over the randomly initialized CNN.

Figure 3: The benefits of unsupervised learning vs. unsupervised to supervised sample ratio. When the ratio is 50:1, we see a 4.09% increase in performance. But the benefit shrinks as the ratio decreases. When the ratio is 1:1, there is a penalty for using unsupervised pre-training.

4.3.1 ADDITIONAL DATA AUGMENTATION: COLOR AND CONTRAST

The current best result on STL-10 (Dosovitskiy et al. (2014)) makes extensive use of additional data augmentation including: scaling, rotation, color and two forms of contrast. They do not perform these augmentations during supervised training, but during a discriminative unsupervised feature learning period. We test the regularizing effects of these additional augmentations when applied directly to supervised training, and test how these regularization effects hold up when combined with with unsupervised pre-training. To do this, we use some of these additional data-augmentations during our supervised training: **color augmentation** and **contrast augmentation**.

Color augmentation: The images are represented in HSV color space (h, s, v). Here we generate a single random number for each image and add it to the hue value for each pixel like so:

$$a \quad \sim \quad Uniform(-0.1, 0.1) \tag{8}$$
$$h \quad = \quad h + a \tag{9}$$

Table 2: Quantitative comparison with other methods on CIFAR-10 (A: Data Augmentation, D: Dropout, U: Unsupervised Learning).

Algorithm	Accuracy
Convolutional Auto-encoders - Masci et al. (2011)	79.20 %
Single layer K-means - Coates et al. (2011)	79.60 %
Convolutional K-means Networks - Coates and Ng (2011)	82.00 %
Exemplar CNN - Dosovitskiy et al. (2014)	82.00 %
Convolutional Kernel Networks - Mairal et al. (2014)	82.18 %
NOMP - Lin and Kung (2014)	82.90 %
Max-Out Networks - Goodfellow et al. (2013b)	90.65 %
Network In Network - Lin et al. (2013)	91.20 %
Deeply-Supervised Nets - Lee et al. (2014)	**91.78 %**
Zero-bias CNN +ADU	86.44 %
Zero-bias CNN +AD	86.70 %

Table 3: Quantitative comparison with other methods on STL-10 (A: Data Augmentation, D: Dropout, C: Color Augmentation, U: Unsupervised Learning).

Algorithm	Accuracy
Convolutional K-means Networks - Coates and Ng (2011)	60.1 % $\pm$ 1.0 %
Convolutional Kernel Networks - Mairal et al. (2014)	62.32 %
Hierarchical Matching Pursuit (HMP) - Bo et al. (2013)	64.5 % $\pm$ 1.0 %
NOMP - Lin and Kung (2014)	67.9 % $\pm$ 0.6 %
Multi-task Bayesian Optimization - Swersky et al. (2013)	70.1 % $\pm$ 0.6 %
Exemplar CNN - Dosovitskiy et al. (2014)	**72.8 % $\pm$ 0.4 %**
Zero-bias CNN +AD	62.01 % $\pm$ 1.9 %
Zero-bias CNN +ADU	65.88 % $\pm$ 0.9 %
Zero-bias CNN +ADC	68.51 % $\pm$ 0.8 %
Zero-bias CNN +ADCU	70.20 % $\pm$ 0.7 %

Contrast augmentation: Here we generate six random numbers for each image, with the following distributions:

$$a, d \quad \sim \quad Uniform(0.7, 1.4) \tag{10}$$
$$b, e \quad \sim \quad Uniform(0.25, 4) \tag{11}$$
$$c, f \quad \sim \quad Uniform(-0.1, 0.1) \tag{12}$$

And use them to modify the saturation and value for every pixel in the image, like so:

$$s \quad = \quad as^b + c \tag{13}$$
$$v \quad = \quad ds^e + f \tag{14}$$

We find that a) additional data-augmentation is incredibly helpful, increasing accuracy by 6.5%, b) unsupervised pre-training still maintains an advantage (1.69%).

5 CONCLUSIONS

We present a new type of convolutional auto-encoder that has zero-bias and ReLU activations and achieves superior performance to previous methods. We conduct thorough experiments on CIFAR-10 to analyze the effects of unsupervised pre-training as a form of regularization when used in isolation and in combination with supervised forms of regularization such as data augmentation and dropout. We observe that, indeed, unsupervised pre-training can provide a large gain in performance when the ratio of unsupervised to supervised samples is large. Finally, we verify our findings by applying our model to STL-10, a dataset with far more unlabeled samples than labeled samples (100:1). We find that with additional regularization, via color augmentation, our method is able to achieve nearly state-of-the-art results.

CODE

All experiments were run using our own open source library Anna, which can be found at: `https://github.com/ifp-uiuc/anna`

Code to reproduce the experiments can be found at: `https://github.com/ifp-uiuc/an-analysis-of-unsupervised-pre-training-iclr-2015`

REFERENCES

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Unsupervised feature learning for rgb-d based object recognition. In *Experimental Robotics*, pages 387–402. Springer, 2013.

Adam Coates and Andrew Y Ng. Selecting receptive fields in deep networks. In *Advances in Neural Information Processing Systems*, pages 2528–2536, 2011.

Adam Coates, Andrew Y Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 766–774. Curran Associates, Inc., 2014.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2013.

Ian J Goodfellow, David Warde-Farley, Pascal Lamblin, Vincent Dumoulin, Mehdi Mirza, Razvan Pascanu, James Bergstra, Frédéric Bastien, and Yoshua Bengio. Pylearn2: a machine learning research library. *arXiv preprint arXiv:1308.4214*, 2013a.

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013b.

Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

Kevin Jarrett, Koray Kavukcuoglu, M Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. *arXiv preprint arXiv:1409.5185*, 2014.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.

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

Tsung-Han Lin and H. T. Kung. Stable and efficient representation learning with nonnegativity constraints. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1323–1331. JMLR Workshop and Conference Proceedings, 2014.

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2627–2635. Curran Associates, Inc., 2014.

Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional autoencoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 52–59. Springer, 2011.

Roland Memisevic, Kishore Konda, and David Krueger. Zero-bias autoencoders and the benefits of co-adapting features. *arXiv preprint arXiv:1402.3337*, 2014.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive autoencoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 833–840, 2011.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 2004–2012, 2013.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Robert Fergus. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE, 2010.

Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2018–2025. IEEE, 2011.