Building Deep Neural Networks

with

Daniel L. Silver, Ph.D.

Andy McIntyre, Ph.D.

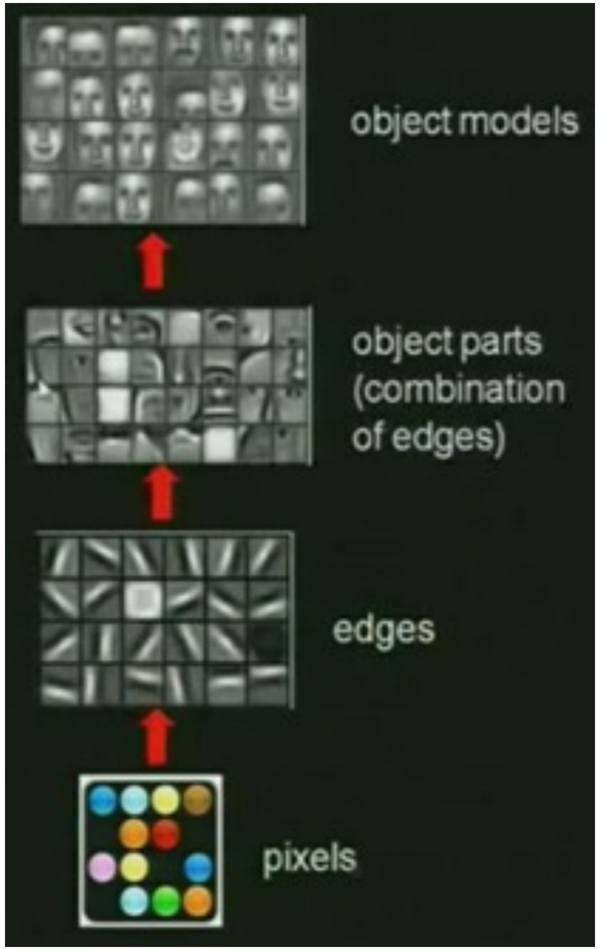
June 24, 2022

Outline

- Background
- Problem of vanishing gradient
  - Beyond two hidden layers
- Methods of overcoming the problem
  - Long patient training
  - Better activation functions
  - Pre-training
  - Constrained internal representation - weight-sharing
  - Better regularizers (sparsity, weight-decay, dropout, injected noise)
- Demo using many layers

Background – Why Deep Networks?

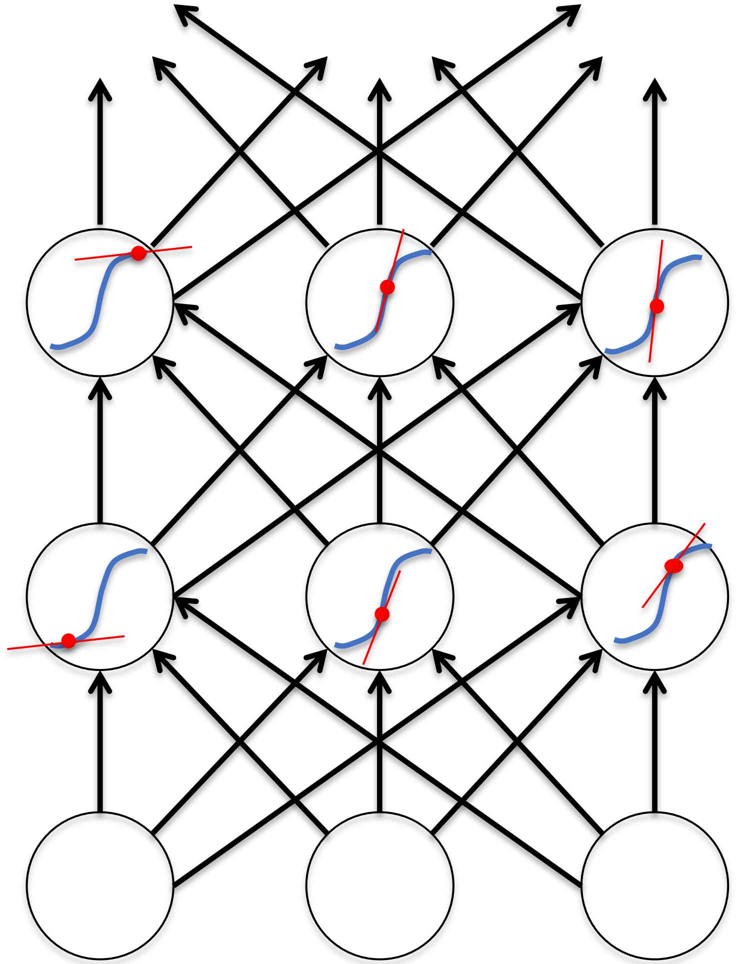
- Accuracy
  - Overcome limitations of shallow networks
  - Upper hidden layers learn higher order features = combinations of lower features
- Applications:
  - Image classification (ImageNet, medical analyses)
  - Voice recognition (Amazon Echo, Google Home)
  - Human competitve results (Game of GO)



Exploding or Vanishing (Unstable) Gradients

- What happens to the magnitude of the gradients as we backpropagate through many layers?
  - If the weights are big the gradients grow exponentially (explode)
  - If the weights are small, the gradients shrink exponentially (vanish)
- Typical feed-forward neural nets can cope with these exponential effects because of few hidden layers
- Deep Networks
  - Exploding gradient - clip the weights
  - Vanishing gradient - bigger problem
  - Why? Check out this video

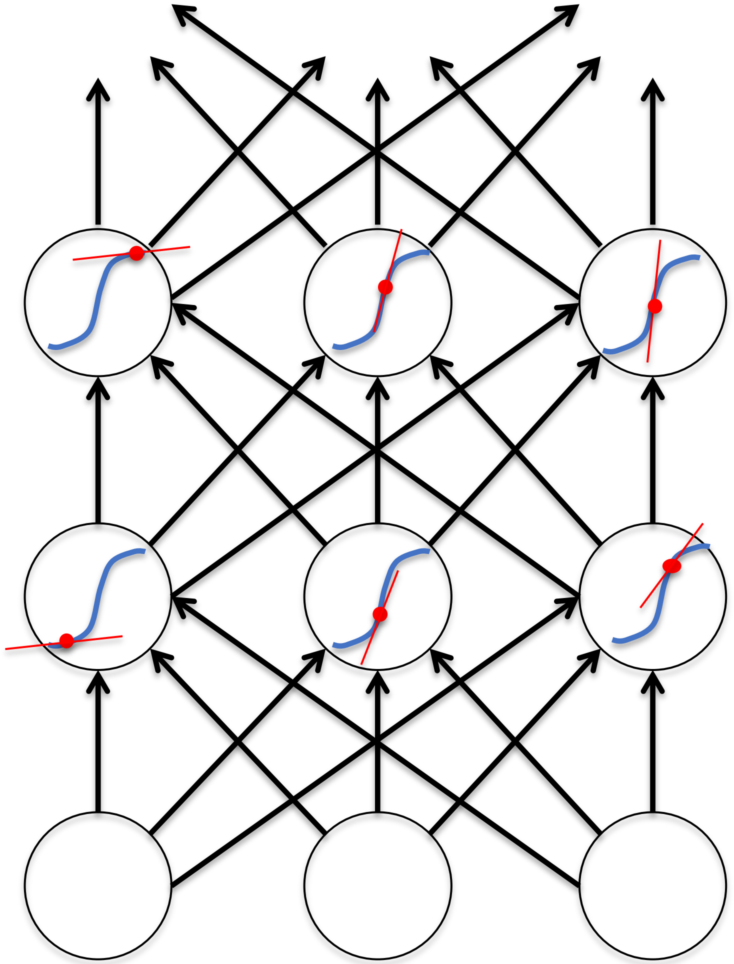
- Details: https://ayearofai.com/rohan-4-the-vanishing-gradient-problem-ec68f76ffb9b



Based on slide by Geoff Hinton

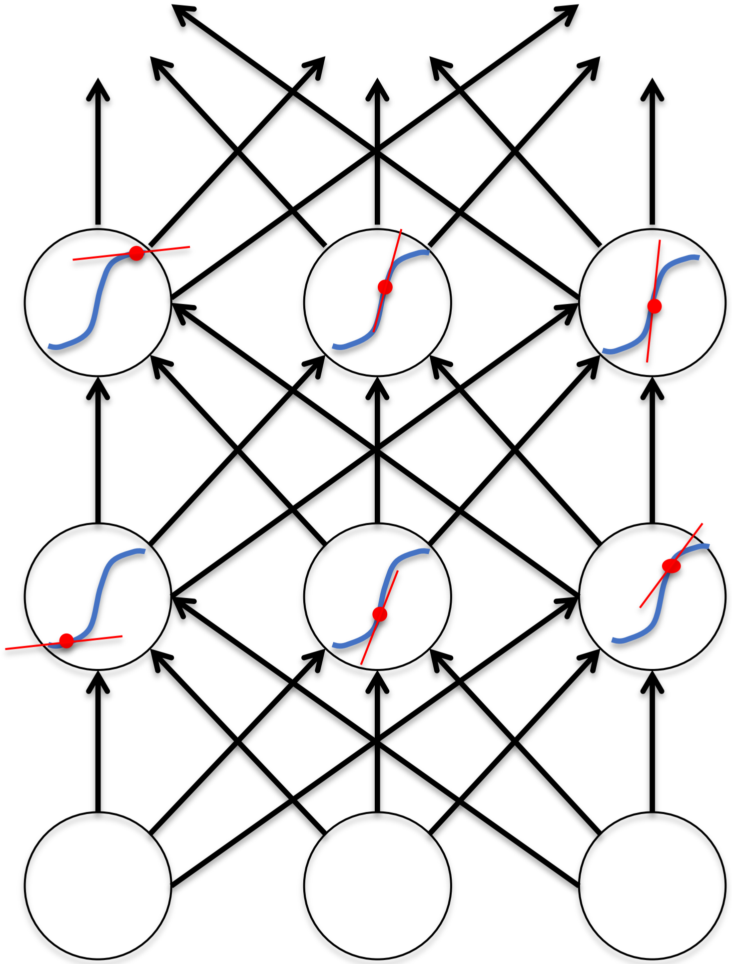
Vanishing Gradient

- Lowest layers of MLP do not get trained well
- Leads to very slow training - especially in lower layers
- Top couple layers can usually learn any task "pretty well”
- The error to lower layers drops quickly
- Lower layers never get the opportunity to use their capacity to improve results, they just create a random feature map
- Need a way for early layers to do effective work

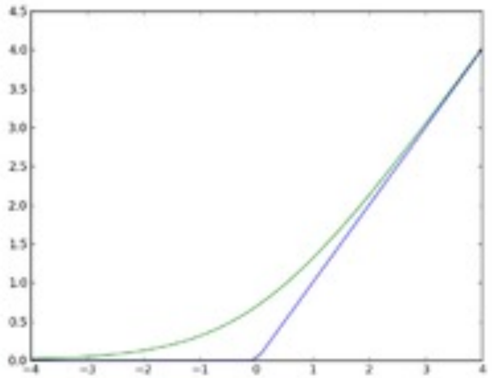


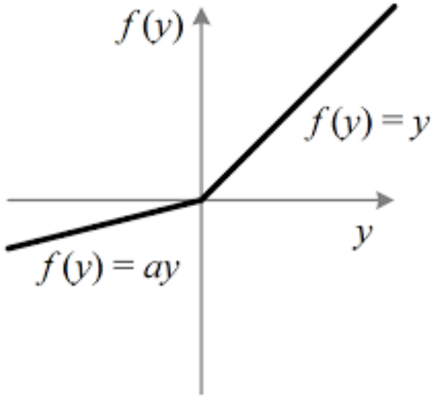
Overcoming Vanishing/Exploding Gradient

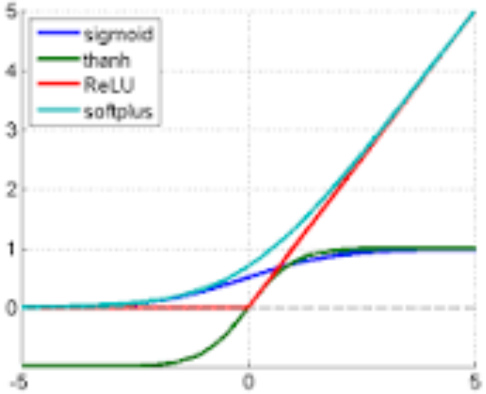
- Better weight initialization
- Perform long patient training
  - small learning rates
- Use parallel processing - GPUs, etc.
- Normalize input to layers
- Use better activation functions
- Regularization through weight control



Better Activation Functions



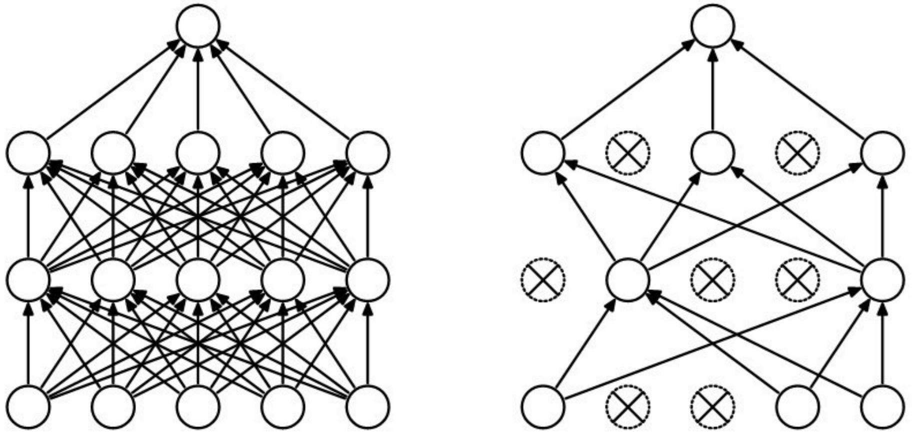




- ***Rectified Linear Units or Relu:*** $f(x) = \text{Max}(0,x)$ - more efficient gradient propagation, derivative is 0 or constant, just fold into learning rate
- **Variations on Relu More efficient computation**: Only comparison, addition and multiplication.
  - Leaky ReLU $f(x) = x$ if $x > 0$ else ax, where $0 \leq a <= 1$, so that derivate is not 0 and can do some learning for net < 0 (does not “die”).
  - Lots of other variations
- **Sparse activation**: For example, in a randomly initialized network, only about 50% of hidden units are activated (have output $\neq 0$)
- Learning in linear range easier for most learning models

Better Regularization of Free Parameters

- Weights (free parameters) make the ANN model
- Beyond the training examples we can control weights
  - Momentum
  - Weight-decay
  - Normalization
  - Sparsity
  - Dropout
  - Injected noise



(a) Standard Neural Net (b) After applying dropout.

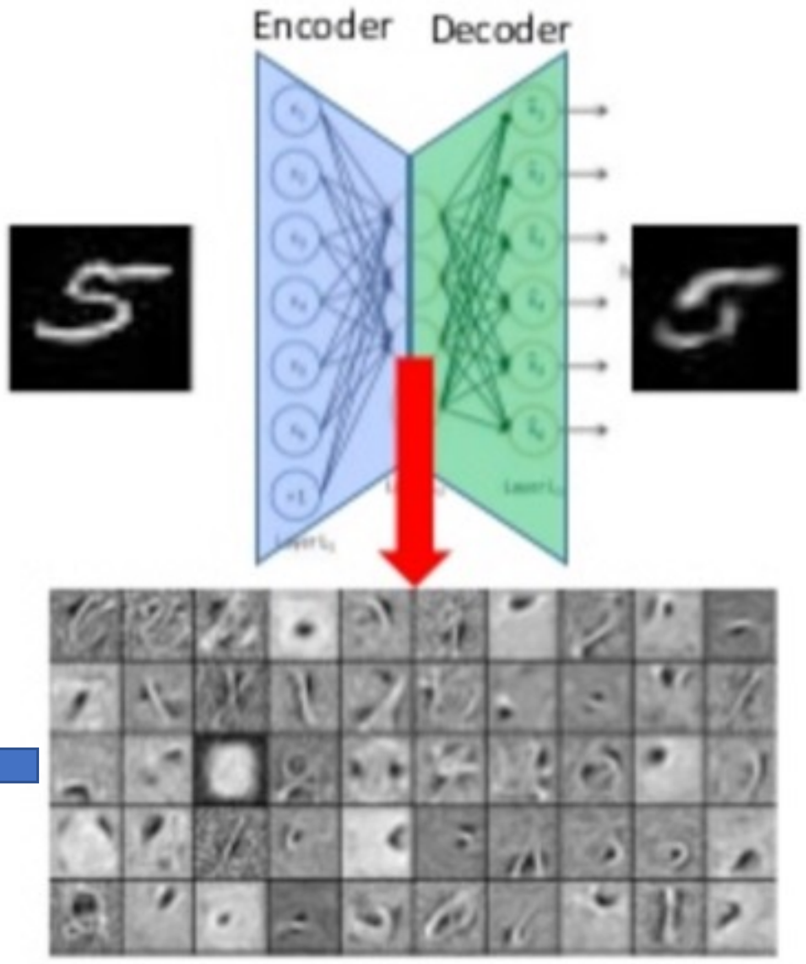
Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Pre-training of Deep Learning Architectures

- Geoff Hinton and Yoshua Bengio (2007+)
  - Learning Internal Representations
  - Layered networks of unsupervised auto-encoders efficiently develop hierarchies of features that capture regularities in their respective inputs
    - Auto-encoders
    - Restricted Boltzmann Machines
  - Can be used to develop models for families of tasks

Autoencoders using BP ANN

- Learn to predict their input at the output
- Learn an internal representation (encoding) of image
- Feature detectors of the input
  - Latent representation
  - Embedding space
- Can be used to pre-train a classifier



Feed a
standard
BP ANN to do
Classification

Deep Learning Architectures

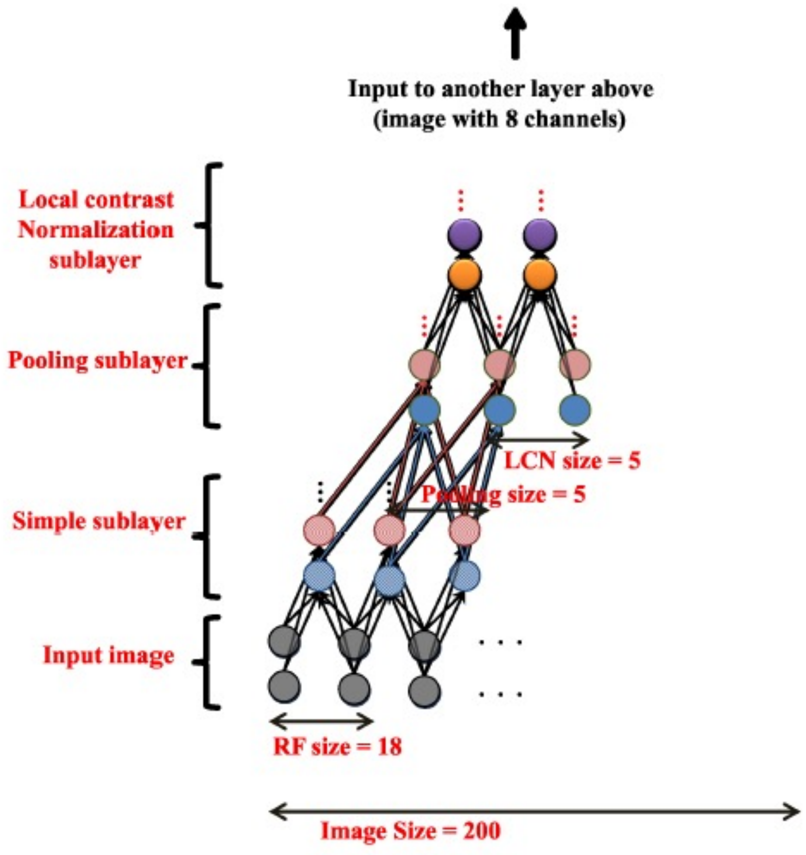
Andrew Ng's work on Deep Learning Networks (ICML-2012)

•Problem: Learn to recognize human faces, cats, etc from unlabeled data

•Dataset of 10 million images; each image has 200x200 pixels

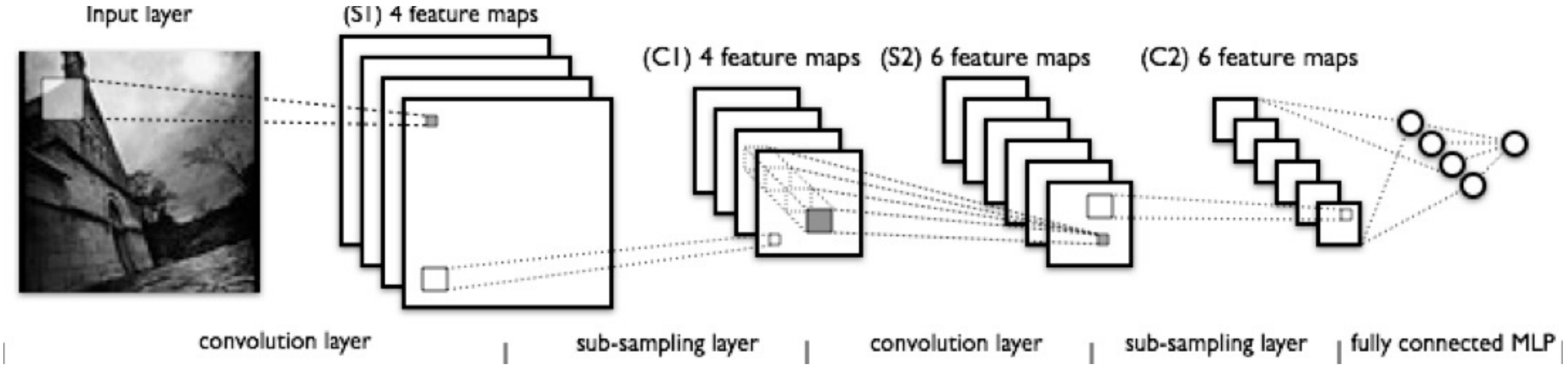
•9-layered locally connected sparse autoencoder neural network (1B connections)

•Parallel algorithm; 1,000 machines (16,000 cores) for three days



Building High-level Features Using Large Scale Unsupervised Learning
Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeffrey Dean, and Andrew Y. Ng
ICML 2012: 29th International Conference on Machine Learning, Edinburgh, Scotland, June, 2012.

Constrained Internal Representation



Weight sharing based on knowledge of the input space

- Reduces overall complexity of model
- One of key hints for using deep learning architectures

TUTORIAL 4

- Demonstrate how the addition of hidden layers makes the development / training of a BP network more challenging
- Try the python code **tf.keras_mnist_deep.ipynb**
- For more, check out the following tutorial: https://www.tensorflow.org/tutorials/keras/classification