BLUE WATERS-GEO

MAPPING AND MODELING THE WORLD

Machine Learning With Distributed Training on Blue Waters

Aaron D. Saxton, PhD, Data Scientist

saxton@illinois.edu

ILLINOIS

NCSA | National Center for Supercomputing Applications

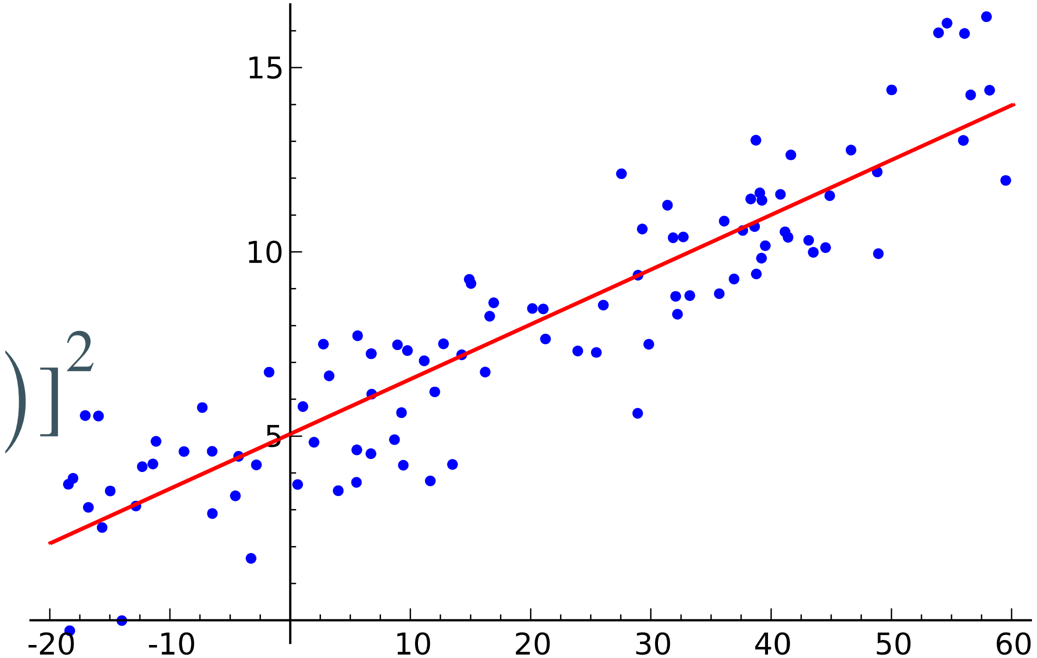
NSF

CRAY

UNCLASSIFIED

Statistics Review

- Simple $y = m \bullet x + b$ regression
  - Goal: Find m,b
  - With data set $\{(x_i, y_i)\}_{i=1,..,n}$
  - Let the error be
  - $$R = \sum_{i=1}^{n} [(y_i - (m \bullet x_i + b)]^2$$
  - Minimize R with respect to m and b.
- In practice we consider more general $y = f(x)$

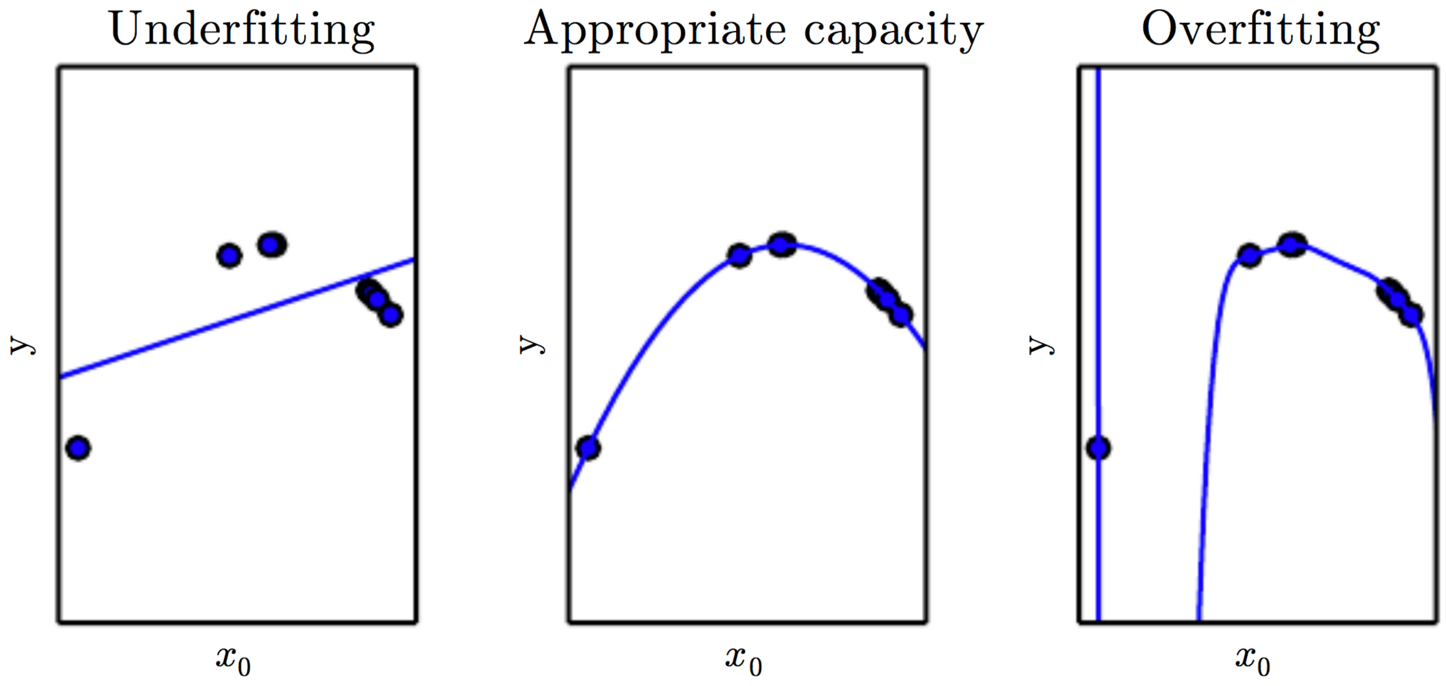


Statistics Review

- Regressions with parameterized sets of functions. e.g.
  - $y = ax^2 + bx + c$ (quadratic)
  - $y = \sum_i a_i x^i$ (polynomial)
  - $y = Ne^{rx}$(exponential)
  - $y = \dfrac{1}{1 + e^{-(a+bx)}}$ (logistic)

Statistics Review

- Rational models of degree 'n'
  - "degrees of freedom"
    - models capacity



Just a Bad Fit

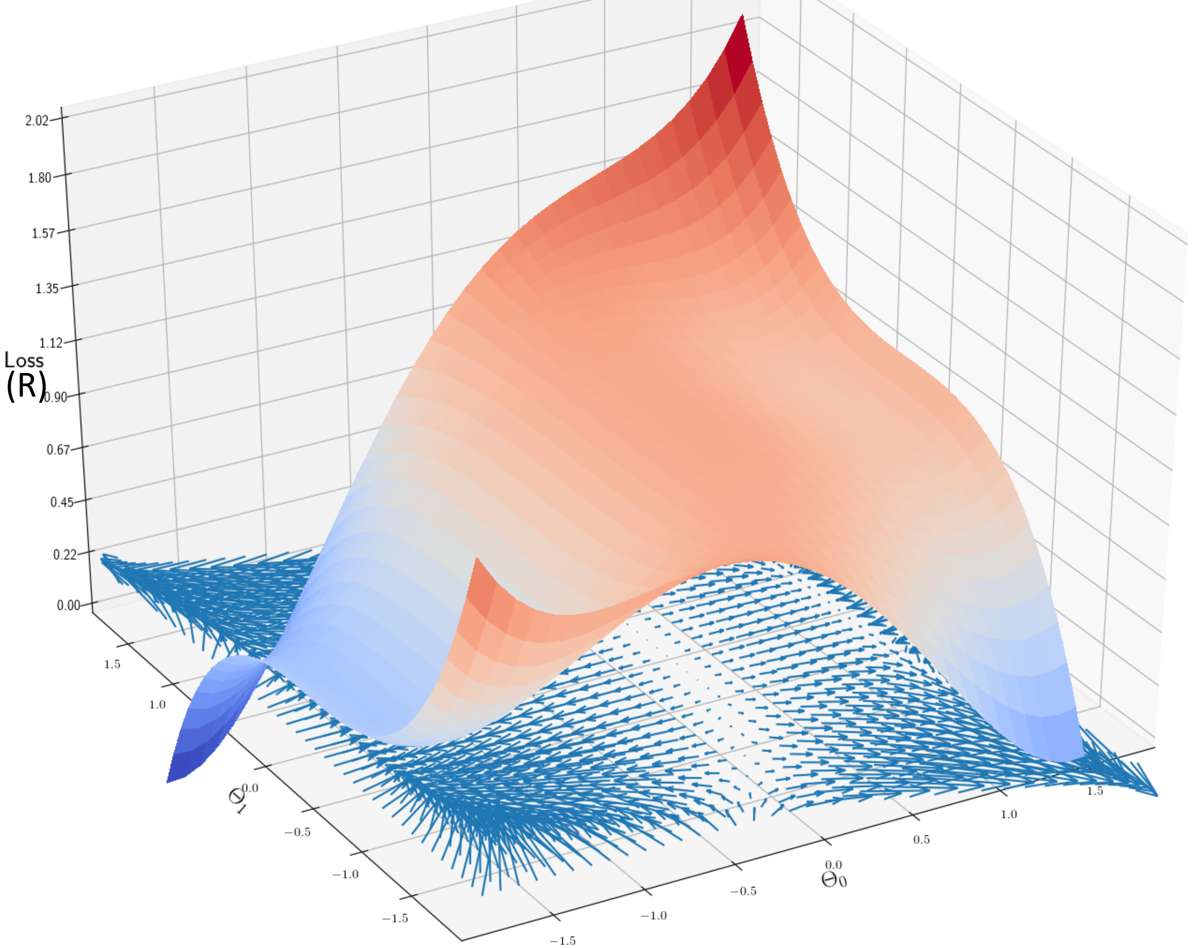
Bad **Generalization**

Deep Learning, Goodfellow et. al., MIT Press, http://www.deeplearningbook.org, 2016

Gradient Decent

- Searching for minimum of
- $R = \sum_{i=1}^{n} [(y_i - f_{\theta_t}(x_i)]^2$
- $\nabla R = \left\langle R_{\theta_0}, R_{\theta_2}, \ldots, R_{\theta_n} \right\rangle$
  - R and ∇R is a sum over i
- Update parameters
  - $R\left(\vec{\theta}_{t+1}\right) = R\left(\vec{\theta}_t + \gamma \nabla R\right)$
  - γ: Learning Rate

Fictitious Loss Surface With Gradient Field

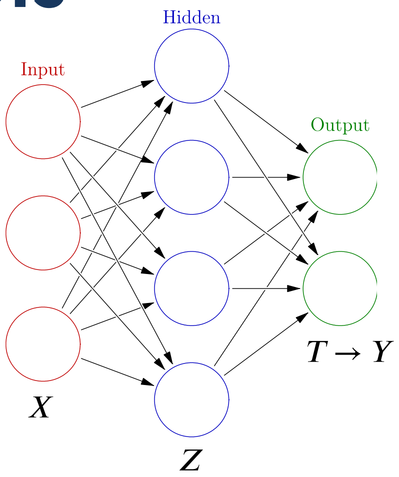


Stochastic Gradient Decent

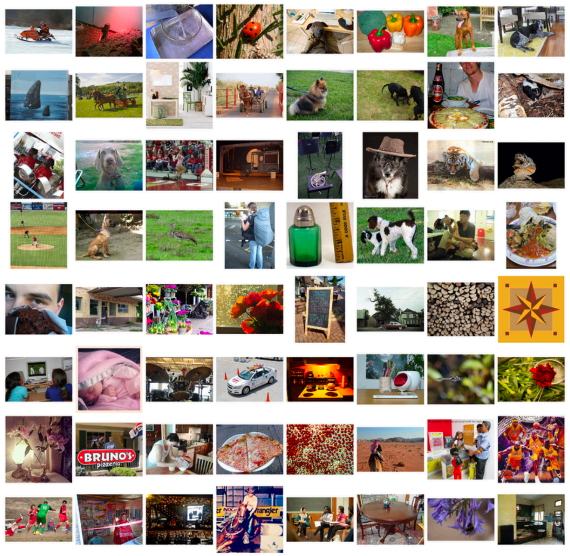
- Single training example, (x_i, y_i), Sum over only one training example
- $\nabla R_{(x_i,y_i)} = \left\langle R_{\theta_0}, R_{\theta_2}, \ldots, R_{\theta_n} \right\rangle_{(x_i,y_i)}$
- $R_{(x_i,y_i)}\left(\vec{\theta}_{t+1}\right) = R_{(x_i,y_i)}\left(\vec{\theta}_t + \gamma \nabla R_{(x_i,y_i)}\right)$
- γ: Learning Rate
- Choose next (x_{i+1}, y_{i+1}), (Shuffled training set)
- SGD with mini batches
- Many training example, (x_i, y_i), Sum over many training example
  - Batch Size or Mini Batch Size (This gets ambiguous with distributed training)
- SGD often outperforms traditional GD, want small batches.
  - https://arxiv.org/abs/1609.04836, On Large-Batch Training … Sharp Minima
  - https://arxiv.org/abs/1711.04325, Extremely Large ... in 15 Minutes

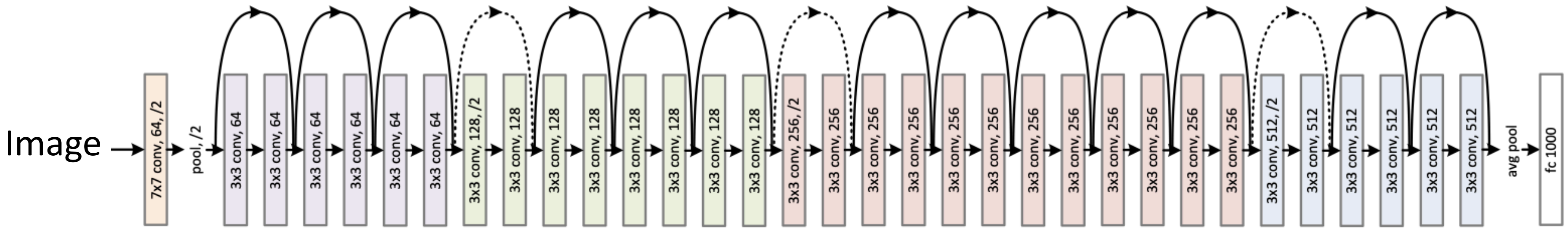
Common ML Models and Datasets

- MNIST
- ImageNet
- Fully connected Neural Network
- Deep Convolutional Neural Network



- $Z_M = \sigma(\alpha_{0m} + \alpha_m X)$
- $T_K = \beta_{0k} + \beta_k Z$
- $f_K(X) = g_k(T)$

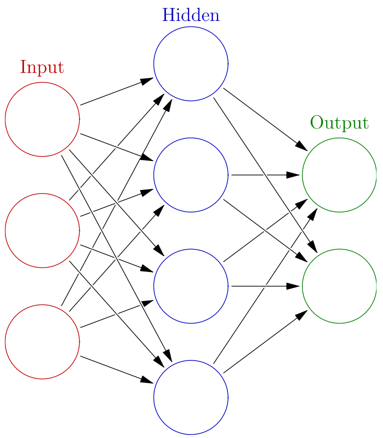


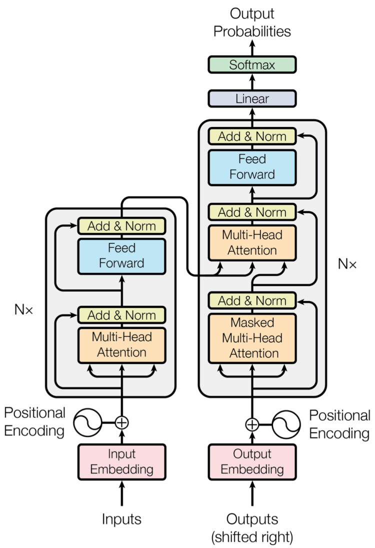


ResNet (34)

Modern ML Taxonomy

- Convolution
- Fully connected layers (traditional “Neural Networks”)
- Max Pooling
- Concatenation
- RNN
- GRU
- Transformers

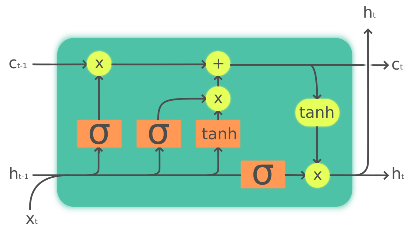




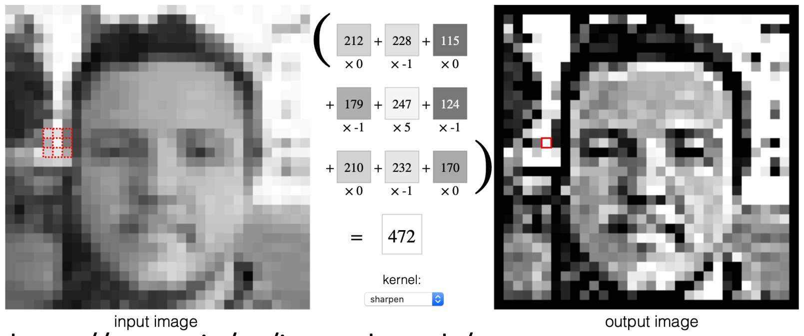
Transformer Architecture
(https://arxiv.org/abs/1706.03762)

sharpen

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

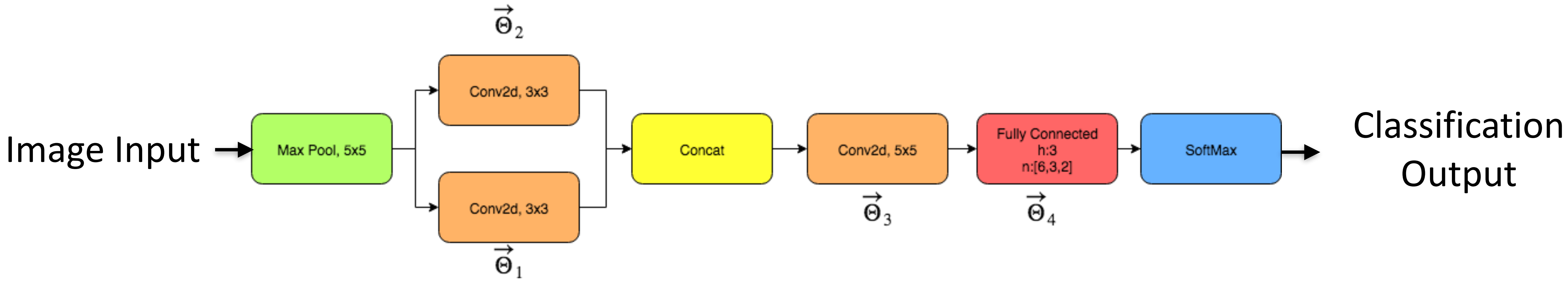


LSTM Cell



https://setosa.io/ev/image-kernels/

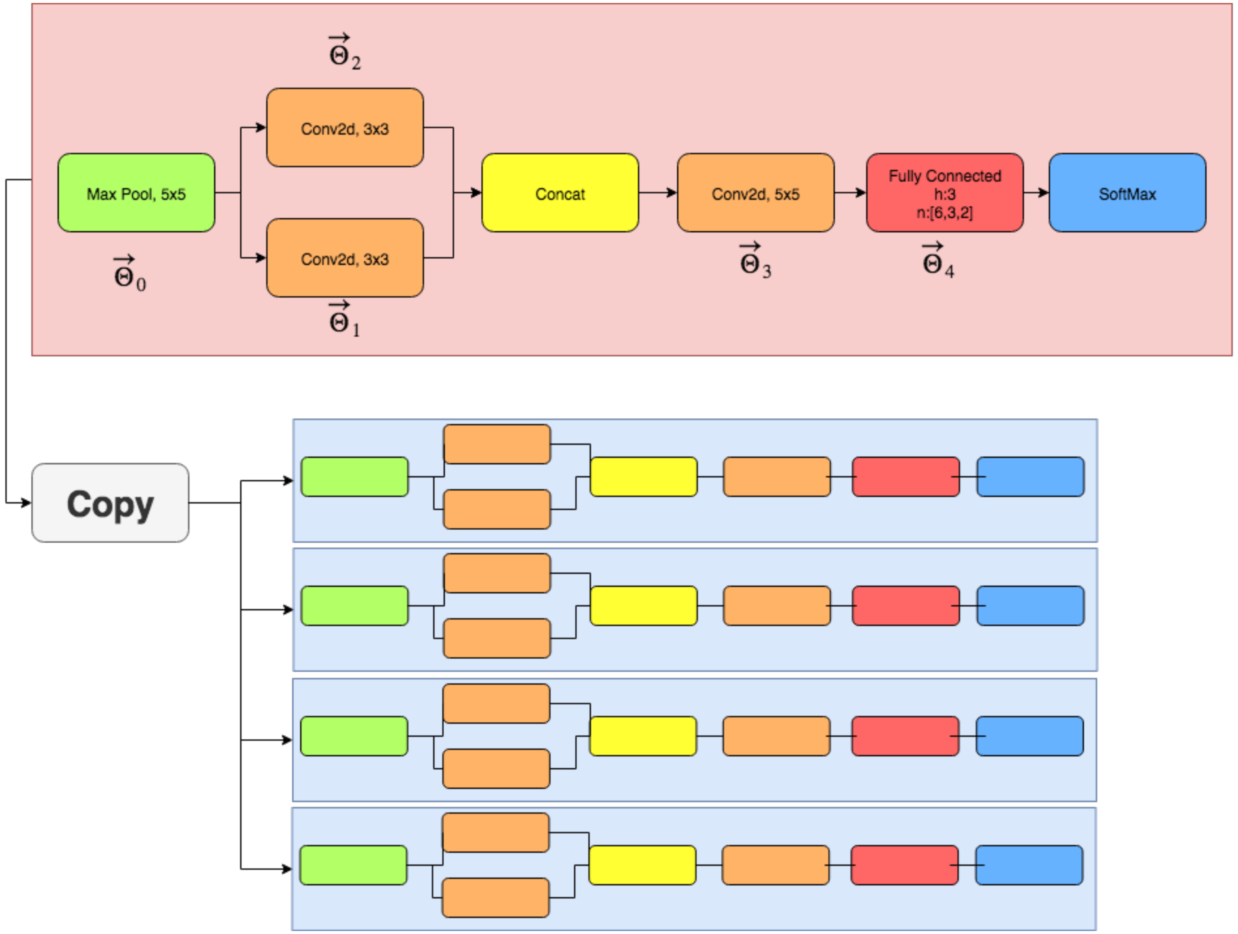
Faux Model Example

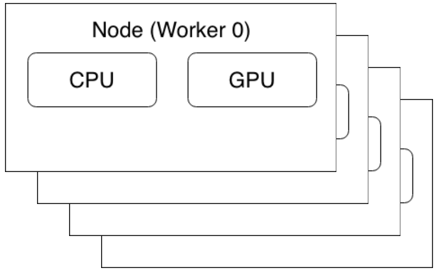


Trainable Weights

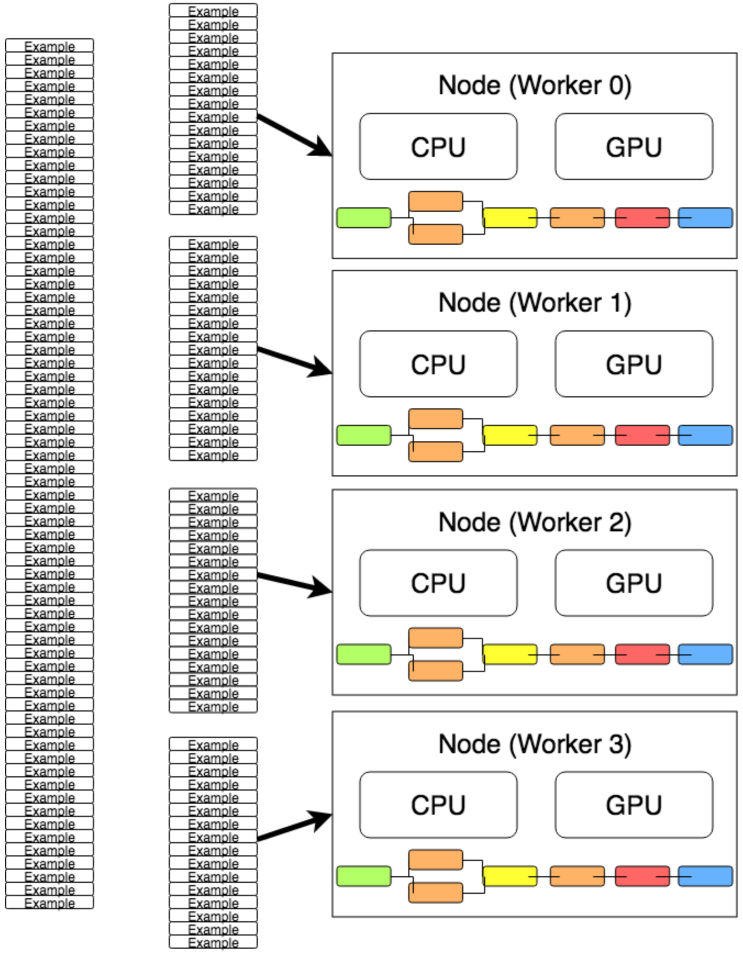
$$\left\{ \vec{\Theta}_i : i \in [0, 1, 2, 3, 4] \right\}$$

Distributed Training, Data Distributed

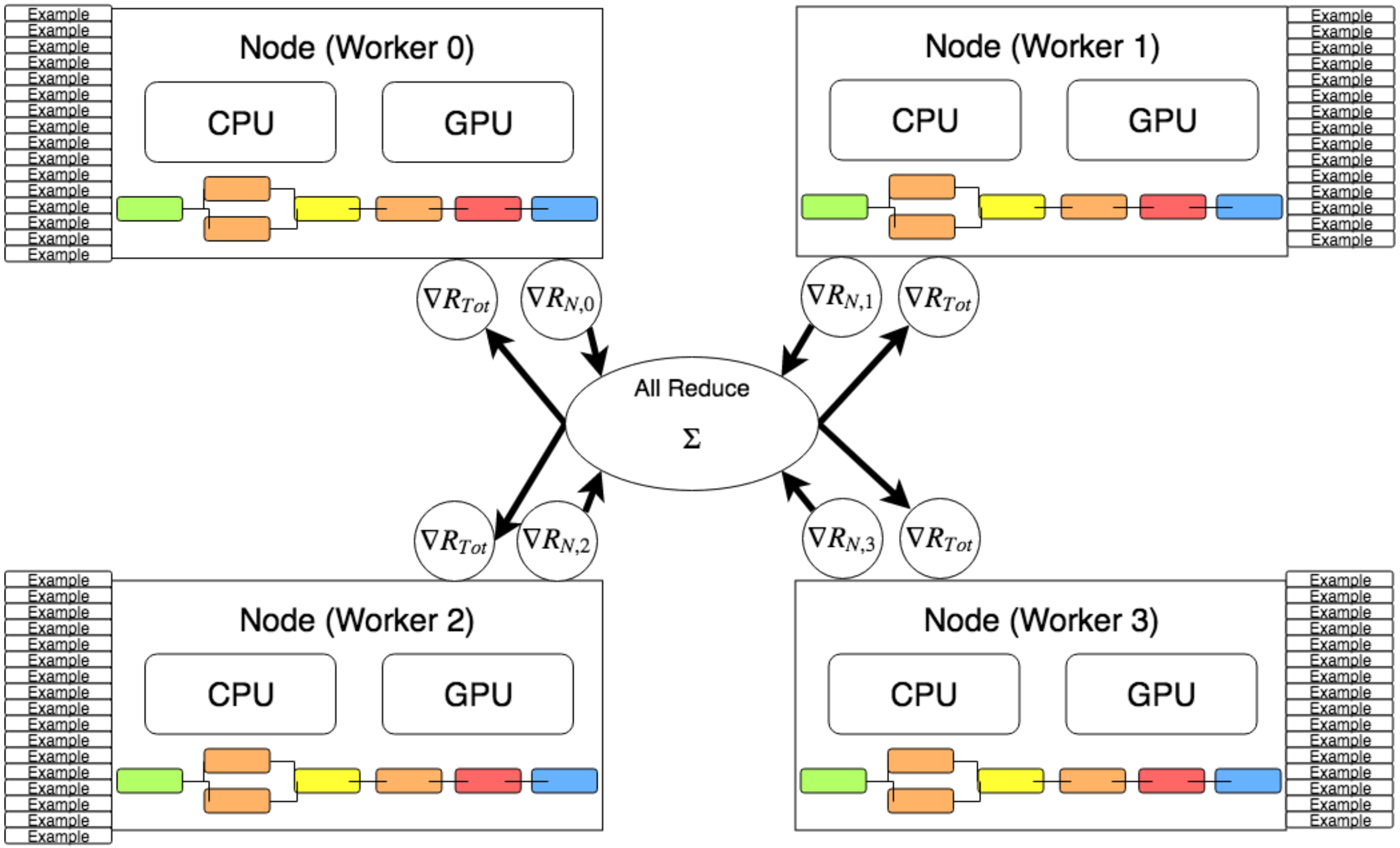




Distributed Training, Data Distributed

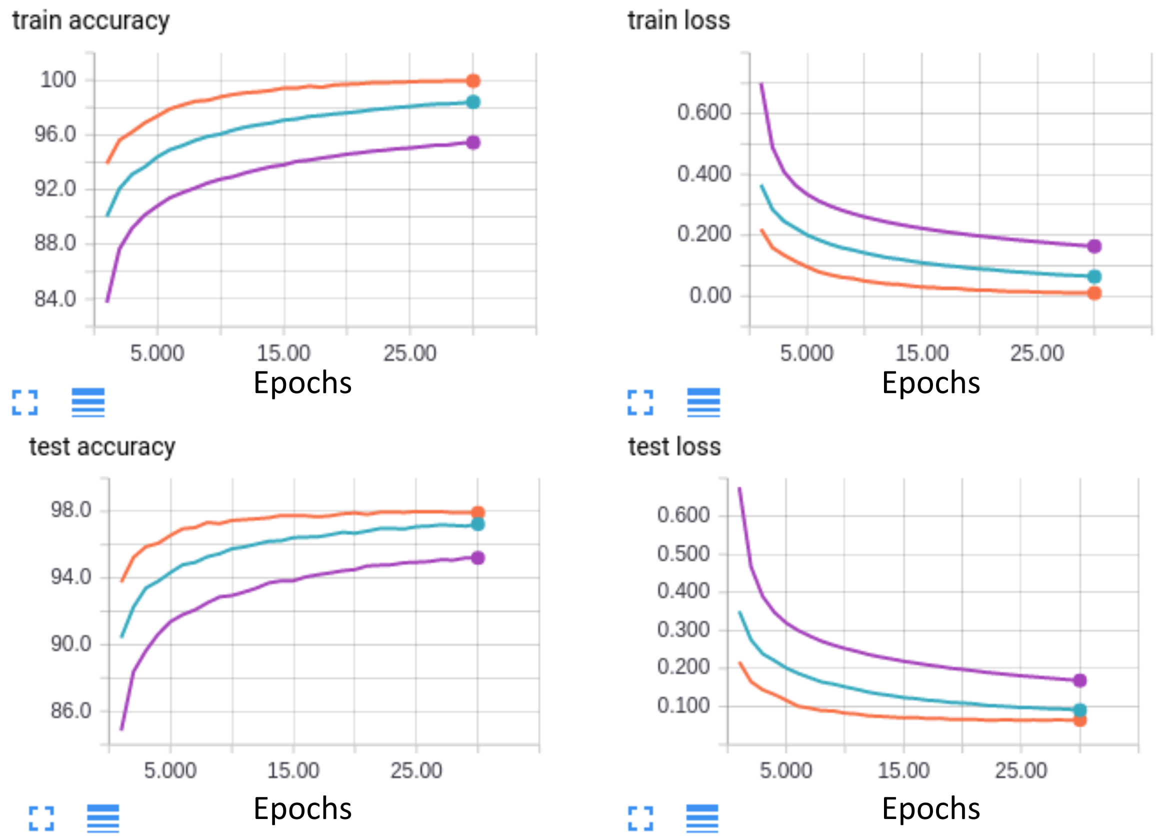


Distributed Training, Data Distributed



Effects of Batch Size on Training, Bad “Generalization”

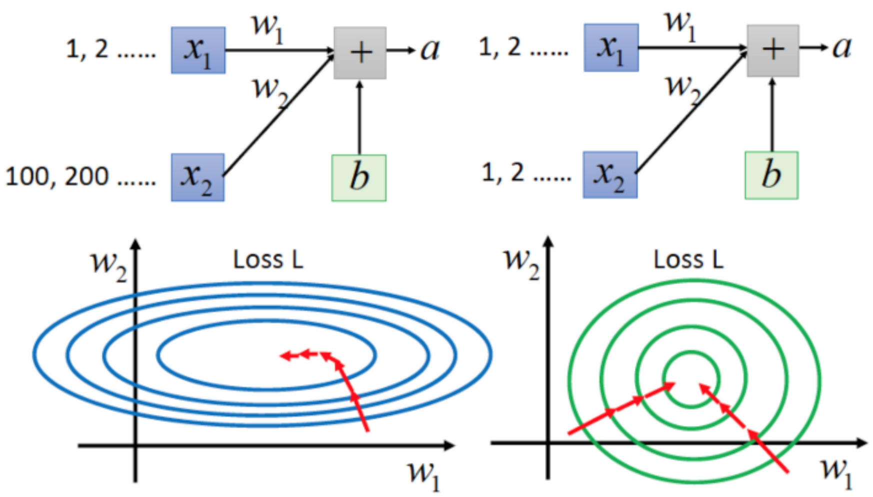
- Training on MNIST dataset
- Model is a NN with 2 FC layers
- Orange: Batchsize 64
  Blue: Batchsize 256
  Purple: Batchsize 1024
- There are ways to mitigate this and scale your model training. Ask me offline: saxton@illinos.edu



https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e

What can we do?

- Regularization!
  - Perturb data
  - Variable learning rate
  - Batch Normalization
  - l1, l2 normalization
  - Dropout
  - Other novel techniques
- No generalized theory that will guarantee what works for you
- Reach out to Aaron Saxton saxton@illinois.edu



[Deep Learning] Batch Normalization
(https://medium.com/@tsengyangyu/batch-normalization-58aac99ee26)

Practical Implementations: Cray ML Plugin

- Cray Optimized MPI Tensor serialization
  - Runs concurrently with standard Tesnorflow

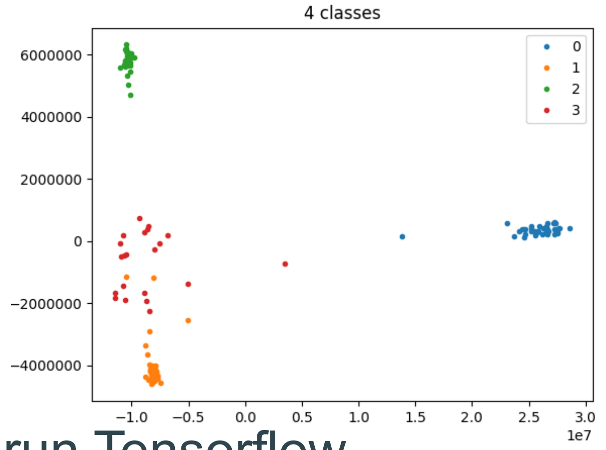
```
import ml_comm as mc
```

Build Data, Model, and Training Somewhere Here

```
tot_model_size = sum([reduce(lambda x, y : x*y, v.get_shape().as_list()) for v in tf.trainable_variables()])
mc.init(1, 1, tot_model_size, "tensorflow")

mc.config_team(0,0,100, FLAGS.num_steps, 2, 1)

class BcastTensors(tf.train.SessionRunHook):
  def __init__(self):
    self.bcast = None

  def begin(self):
    new_vars = mc.broadcast(tf.trainable_variables(), 0)
    self.bcast = tf.group(*[tf.assign(v, new_vars[k]) for k, v  in enumerate(tf.trainable_variables())])
grads_and_vars = optimizer.compute_gradients(total_loss)
grads = mc.gradients([gv[0] for gv in grads_and_vars], 0)
gs_and_vs = [(g,v) for (_,v), g in zip(grads_and_vars, grads)]

train_op = optimizer.apply_gradients(gs_and_vs, global_step=global_step)
hooks = [tf.train.StopAtStepHook(last_step=FLAGS.num_steps), BcastTensors()]
```

```
with tf.train.MonitoredTrainingSession(checkpoint_dir=FLAGS.checkpoint_dir,
                                       save_summaries_steps=20,
                                       save_checkpoint_secs=120,
                                       config=config,
                                       hooks=hooks) as mon_sess:
  print("worker %s: In MonitoredTrainingSession() context" % rank)
  tf.train.start_queue_runners(sess=mon_sess)
```

Practical Implementations: A case study in unsupervised ML on BW



- Unsupervised timeseries classification
  - Novel combination of LSTM, Auto Encoder, K-Means
- Hosting 3TB of time series data in memory
  - Distributed, Indexed, and Queryable: **MongoDB**
- **Heterogenous Jobs** using XE nodes to host MongoDB cluster XK nodes to run Tensorflow
  - ~3000 XE nodes
  - 64 XK nodes
- Training unsupervised models on data samples that range from ~100MB to ~10GB

- Diminishing return on time to solution at 64 XK nodes
  - Personal observation that most vanilla models give fastest time to solution between with batch size between 16 and 64.

BLUE WATERS-GEO

MAPPING AND MODELING THE WORLD

Questions?

Machine Learning With Distributed Training on Blue Waters

Aaron D. Saxton, PhD, Data Scientist

saxton@illinois.edu