Training

Arthur J. Redfern
axr180074@utdallas.edu
Oct 08, 2018
Oct 10, 2018

Outline

- Motivation
- Data
- Initialization
- Forward pass
- Error calculation
- Backward pass
- Weight update
- Evaluation
- Hyper parameter selection
- References

The 1 learning hypothesis of the brain …

C. Metin and D. Frost, "Visual responses of neurons in somatosensory cortex of hamsters with experimentally induced retinal projections to somatosensory thalamus," PNAS Neurobiology, p. 357-361, 1986.
-> Taught the somatosensory cortex to see in hamsters

A. Roe et. al., "Visual projections routed to the auditory pathway in ferrets: receptive fields of visual neurons in primary auditory cortex," Journal of Neuroscience, p. 3651-3664, 1992.
-> Taught the auditory cortex to see in ferrets

Motivation

Parameter Estimation To Maximize Accuracy

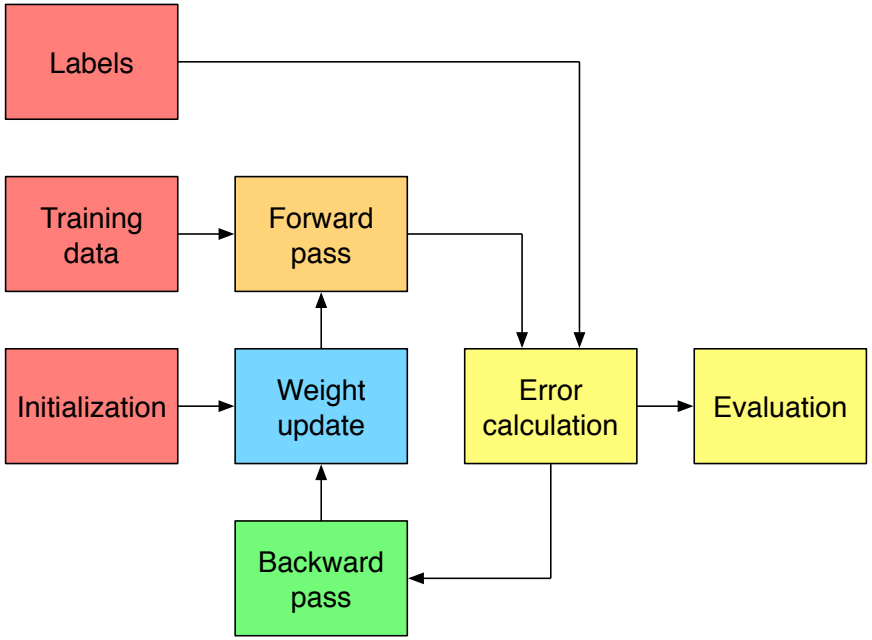
- The previous lectures looked at a lot of CNN designs
- The goal of training is to find the parameters that control the input output mapping of the (typically) linear operations in a CNN design to maximize the accuracy of the full network on testing data
- Conceptually, extracting information from training data to enable the extraction of information from testing data

Example of doing this as a human in this class

- We look at a lot of different networks, I label the different parts and describe how they work together to allow the network to do what it does
- From this you start to build knowledge in network design (training)
- You read a new paper with a new network design
- In the new paper you recognize the different parts and recognize new innovations and how they interact to allow the network to do what it does (testing)

Supervised Learning Framework

- Hyper parameter selection
- Initialization
- Training
  - Update (serial or parallel)
    - Training data selection
    - Forward pass
    - Error calculation
    - Backward pass
    - Weight update
    - Repeat (~ batch)
  - Evaluation / validation
    - Validation data accuracy
    - Break if appropriate
  - Repeat (~ epoch)
- Testing
  - Evaluation / testing
    - Testing data accuracy

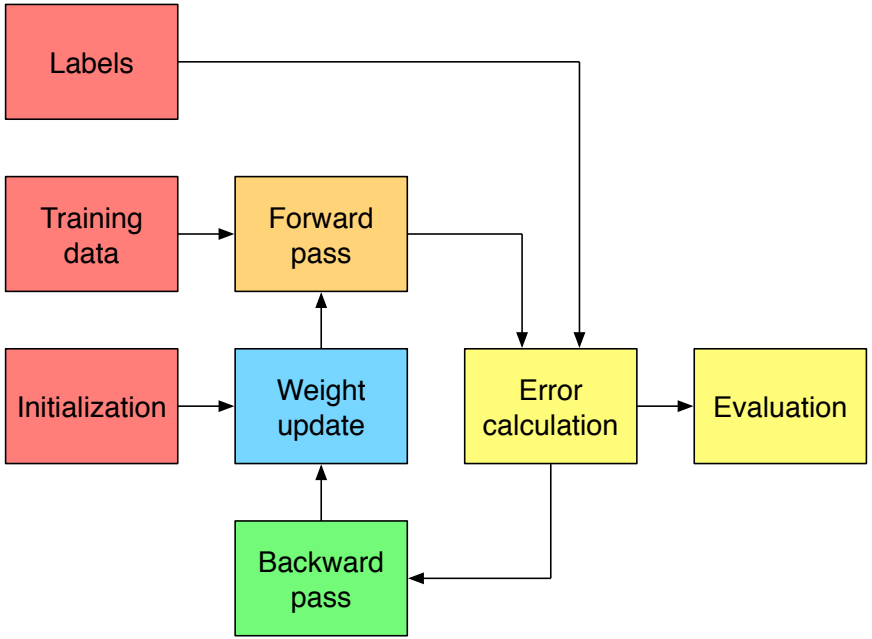


We've seen much of this already

CNN Training Vs Function Optimization

Training builds on the optimization framework discussed during the calculus lectures but there are meaningful conceptual and practical differences

- Differences
  - Different data sets
  - Different initialization strategies
  - Forward pass modifications
  - Different error calculations
  - Backward pass modifications
  - Different weight updates
  - …
- Will look at each of these in subsequent slides



Overfitting Regularization And Generalization

All definitions are informal; see https://developers.google.com/machine-learning/glossary/ for a general glossary of terms

- Generalization is the ability of a model (network) to make accurate predictions on testing data given training on training data
  - Generalization gap is the difference in accuracy of the network on training data vs testing data
  - Differences between training and optimization lead to a potential issue with generalization

- Overfitting refers to a model optimized on training data in a way that fails to nicely generalize to testing data
  - CNNs are highly parameterized models that while excellent in their universal approximation capabilities have the downside of potentially overfitting training data

- Regularization modifies training to improve generalization
  - A number of different regularization strategies will be described in subsequent sections

Overfitting Regularization And Generalization

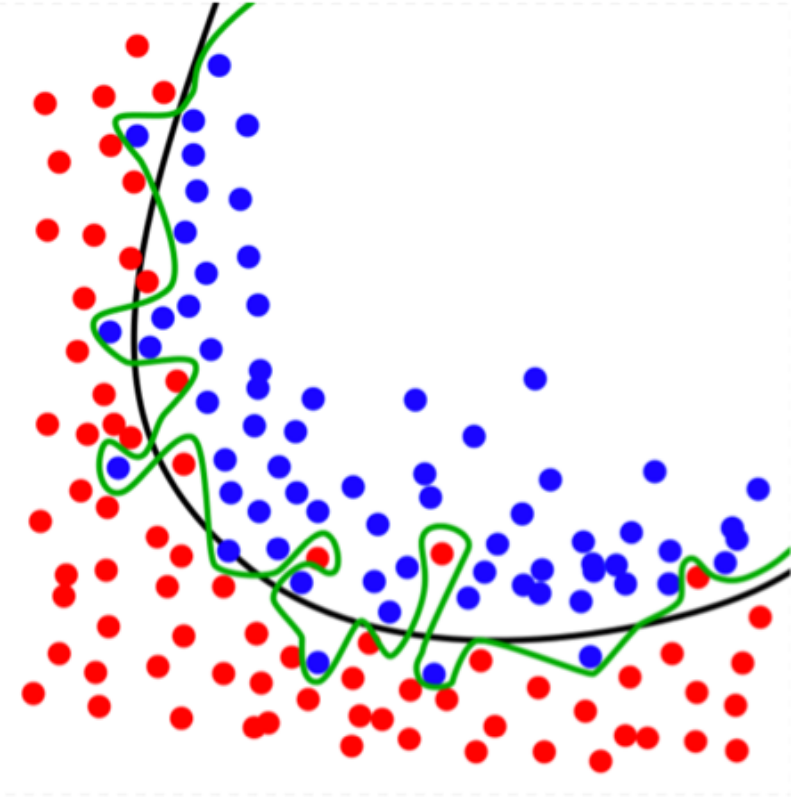


Figure from https://en.wikipedia.org/wiki/Overfitting#/media/File:Overfitting.svg

Overfitting Regularization And Generalization

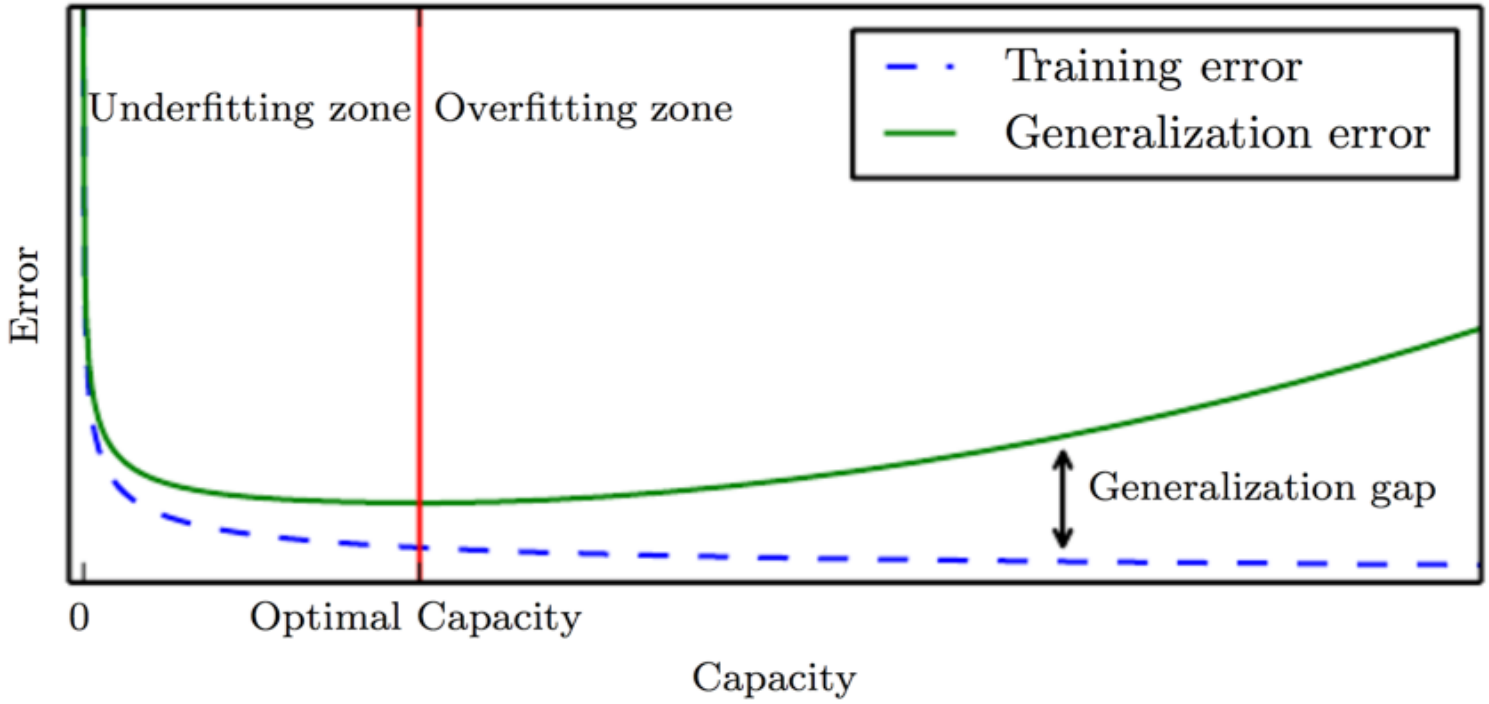


Figure from https://www.deeplearningbook.org

Convergence And Generalization

- Given the previous slides, on subsequent slides you will notice methods serve 1 or more purposes
  - Improve convergence
  - Improve generalization

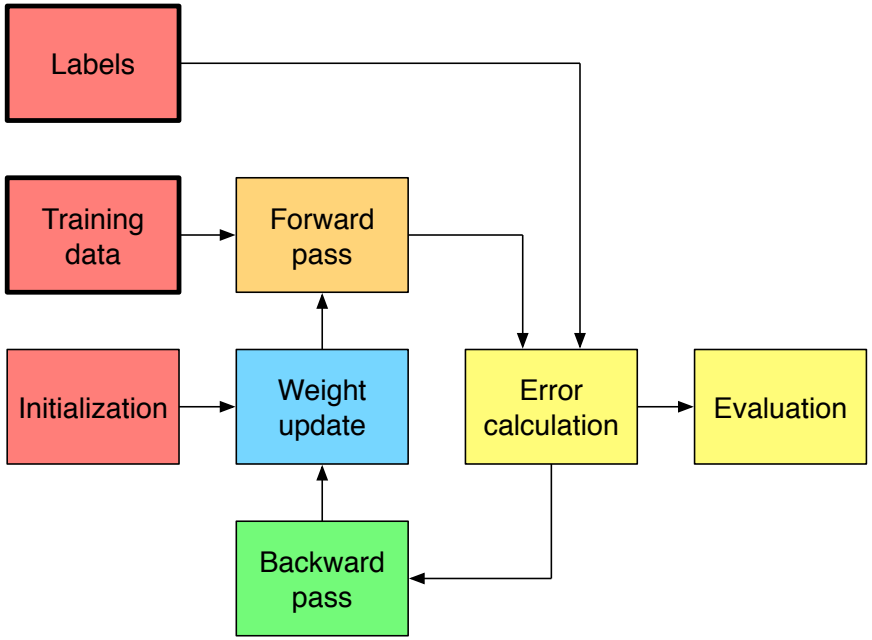
- Example: to improve convergence in very deep networks
  - Use optimal standard deviation during weight initialization
  - Add batch norms after convolutions
  - Use residual connections

- Keep this in the back of your mind as you continue reading

Data

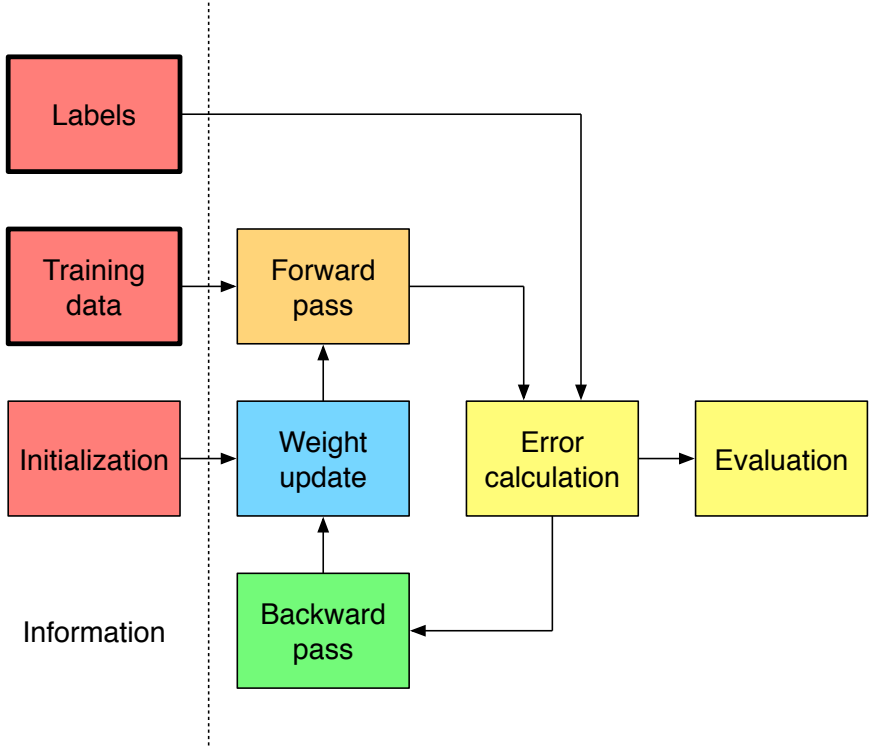
CNN Training Vs Function Optimization

- CNN training
  - 1 data set used for training
  - Typically a different data set used for validation
  - Definitely a different data set used for testing
- Function optimization
  - Same data set used for training and optimization
  - Generalization is less of / not an issue



Data Contains Information

- Impossible to overstate the importance of data (really information) to learning
- For training
  - The more data the better
  - Want good representations of all possible classes / types
  - Want in as many settings as possible
  - Want as similar to the testing data as possible
- Note that initialization weights potentially also contain information (they will be discussed in the next section)

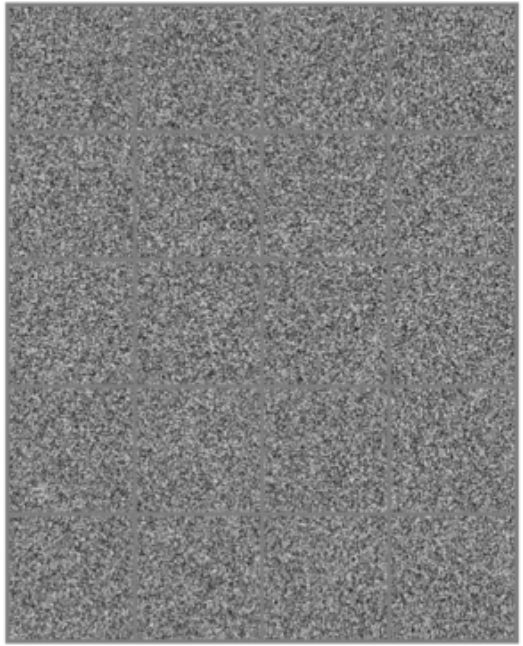


The Curse Of Dimensionality

- The curse of dimensionality: available data for training is sparse relative to all possible data realizations
  - Consider images
  - Consider sounds

- But we can train networks using sparse training data to work on testing data for many cases that we care about: how is this possible?
  - Natural <images, sounds, ...> live on a much smaller dimensional subspace than all possible
  - Successful applications of machine learning are likely possible because of this
  - Frequently exploited via some time, space, spatial frequency, ... in the data

20 random images
(that look nothing like natural images)



Question: how many different possible 8 bit images of size 3 x 1024 x 2048 are there? Note: $256^{6000000}$ is a big number

Figure from https://www.deeplearningbook.org

Training | Validation | Testing Data Split

- Reminder: training data ∩ testing data = ∅
  - Train on training data
  - Test on testing data
- Training and validation data
  - Training data is used for weight updates
  - Validation data is used to periodically monitor progress during training
  - Typical strategy: split training data into training data and validation data
  - Variations on a theme (amount of split, what data to include where, ...)
- Testing data
  - Use to estimate final performance
  - Hidden danger: repeated passes through the flow make effectively make testing data part of training data
  - This is a very real problem for xNNs with lots of parameters and testing many different network and / or training configurations

CNNs vs other ML methods

- It's common in many machine learning training methods to have a small amount of training data and do cross validation (over repeated trials choose different partitions of the training data for training and validation)
- 1 problem with this for CNNs is that CNNs typically don't train well with a small amount of data
- Another problem is that the high capacity of CNNs makes memorization possible which hurts the use of validation data for determining when to stop training
- This is related to the hidden danger mentioned under testing data

2 Classes Of Training Data

Natural

Synthetic

Figures from https://arxiv.org/abs/1605.06457

Natural Data

- Natural data takes advantage of nature's data generation process
  - Doesn't require information on our part
  - Nature supplies the intelligence
  - Natural as used here applies to data generated or measured from a physical environment by a person
  - Ex: images, sounds, radar, lidar, ultrasound, EEG, ...

- It's nice if there's relative uniformity of data generated by different sensors of the same type
  - This allows for training data taken from 1 sensor to be used to train weights for a network that processes a different sensor
  - It typically implies that the data can it be transformed to a common representation (e.g., image sensor and an ISP)
  - Different sensor types can have different common representation (e.g., maybe a RGB image for an image sensor vs a point cloud for a radar)

- A challenge is labeling
  - The number of samples needed for training large network is large
  - Potentially the complexity of labeling even a single sample is high

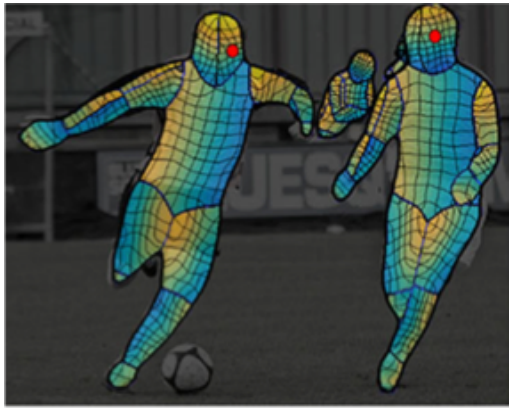
Labeling Data (Is Miserable)

- CNNs like lots of data for training
- The complexity of labeling a single sample can vary a lot
  - Easy
    - 1 input 1 label a few classes
  - Moderate
    - 1 input 1 label many classes
  - Hard
    - 1 input many labels
    - Common to build tools to help
    - Streamline the labeling process
  - Impossible (-ish)
    - Depth after the fact

Figures from https://arxiv.org/abs/1604.01685

Labeling Data (Is Still Miserable)

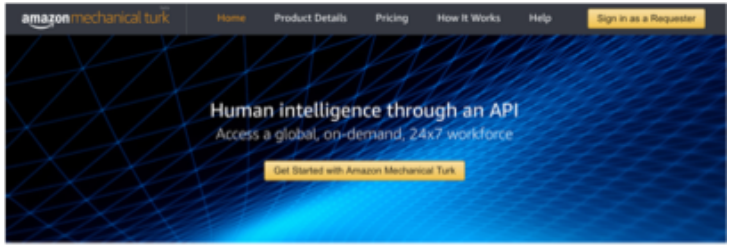
- CNNs like lots of data for training
- The complexity of labeling a single sample can vary a lot
  - Easy
    - 1 input 1 label a few classes
  - Moderate
    - 1 input 1 label many classes
  - Hard
    - 1 input many labels
    - Common to build tools to help
    - Streamline the labeling process
  - Impossible (-ish)
    - Depth after the fact



Figures from https://research.fb.com/facebook-open-sources-densepose/

Tools Money Deception And Coercion

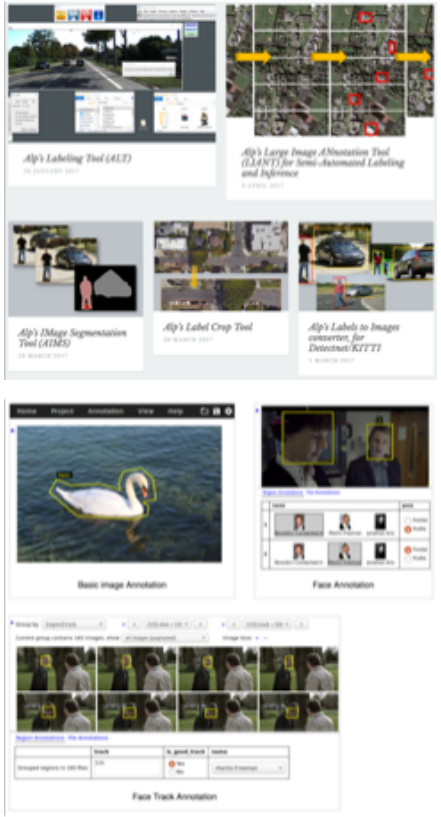
- In practice, build tools then use tools + humans to label
- How to get humans to label
  - Pay people (Mechanical Turk)
  - Coerce people (grad students, relatives)
  - Figure out a way of getting people to do it even though they don't realize they are (games, apps)
  - Realize it's not getting done as fast as you'd like and do it yourself



Amazon Mechanical Turk (MTurk) operates a marketplace for work that requires human intelligence. The MTurk web service enables companies to programmatically access this marketplace and a diverse, on-demand workforce. Developers can leverage this service to build human intelligence directly into their applications.

While computing technology continues to improve, there are still many things that human beings can do much more effectively than computers, such as identifying objects in a photo or video, performing data de-duplication, transcribing audio recordings or researching data details. Traditionally, tasks like this have been accomplished by hiring a large temporary workforce (which is time consuming, expensive and difficult to scale) or have gone undone.

MTurk aims to make accessing human intelligence simple, scalable, and cost-effective. Businesses or developers needing tasks done (called Human Intelligence Tasks or "HITs") can use the robust MTurk API to access thousands of high quality, global, on-demand Workers—and then programmatically integrate the results of that work directly into their business processes and systems. MTurk enables developers and businesses to achieve their goals more quickly and at a lower cost than was previously possible.



Figures from https://www.mturk.com, https://alpslabel.wordpress.com and http://www.robots.ox.ac.uk/~vgg/software/via/

The Dream

- The dream: training CNNs with unlabeled data (unsupervised learning)
  - There's usually orders of magnitude more unlabeled data than labeled data
- The reality: training CNNs tends to work a lot better with labeled data (supervised learning)
- Likely some tradeoff / information balance between unsupervised and supervised learning
  - Less data (information) + labels (information)
  - More data (information) but no labels

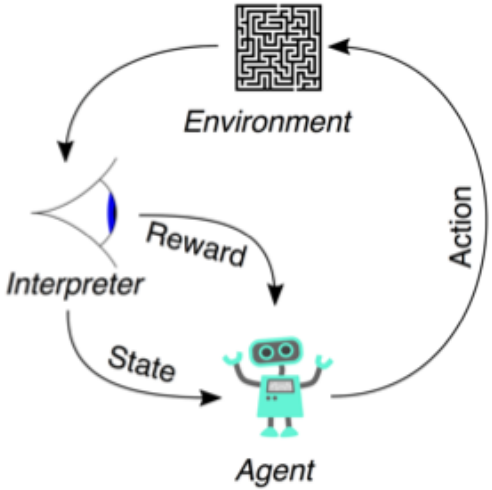
Figures from https://www.mturk.com, https://alpslabel.wordpress.com and http://www.robots.ox.ac.uk/~vgg/software/via/

Somewhere In Between ...

- ... unsupervised and supervised learning lies ...

- Semi supervised learning (some labeled data, some unlabeled data)
  - Example: train multiple CNNs with labeled training data using supervised learning
  - Use unlabeled data as an input and treat the ensemble output label as the correct label
  - Use the ensemble output label with supervised learning to update the individual CNNs

- Reinforcement learning
  - From the current state an agent chooses an action and the environment provides a reward and new state; the combination of input {current state, action} and output {reward, new state} can be used to generate an error signal
  - But the output is frequently sort of 1 step removed from a label (e.g., what really is the value of the new position); so the information content in the output is typically not as strong as the supervised learning cases
  - As such, most reinforcement successes are linked to cases where huge amounts of simulated input output pairs are possible to test and smart strategies are used to approximate labels



Note: will discuss reinforcement learning more in the context of games

Figure from https://upload.wikimedia.org/wikipedia/commons/1/1b/Reinforcement_learning_diagram.svg

Cleaning

- If data is incorrectly labeled then training on it can negatively impact testing accuracy
- Cleaning: remove bad data
- Examples
  - Multiple people labeling the same data
  - 0 lag tick filtering (reminder: tell story)

$$E = mc^3$$

(if you were a physics student and learned this it would likely negatively impact your testing accuracy, i.e., grade; xNNs are no different)

Examples

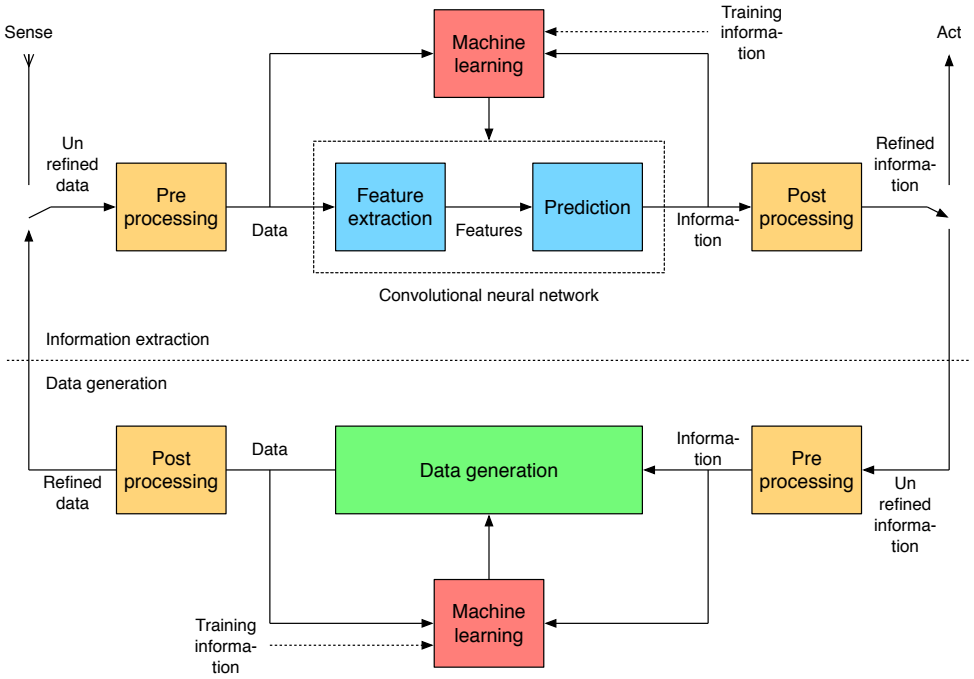
- Labeled classification datasets were created 1st for vision
  - Easiest task to outsource (e.g. Amazon Mechanical Turk)
  - Largest public dataset: ImageNet (1.2M images, 1000 classes)
  - Largest private dataset (that I know about): JFT (100M images, 15k classes)
- Pixel wise labeling can be very expensive
  - Segmentation example: Daimler Cityscapes
    - Fine annotations (5,000 images @ 90+ min each)
    - Coarse annotations (20,000 images @ 7 min each)
    - Total time: 9833 hours (24.5 weeks w / 10 full time people)
  - Other examples: depth, motion, ...
- Examples datasets for vision
  - MNIST, CIFAR 10 / 100, ImageNet, Pascal VOC, KITTI, COCO, Cityscapes, ...

Images from Microsoft COCO
http://cocodataset.org/

Synthetic Data

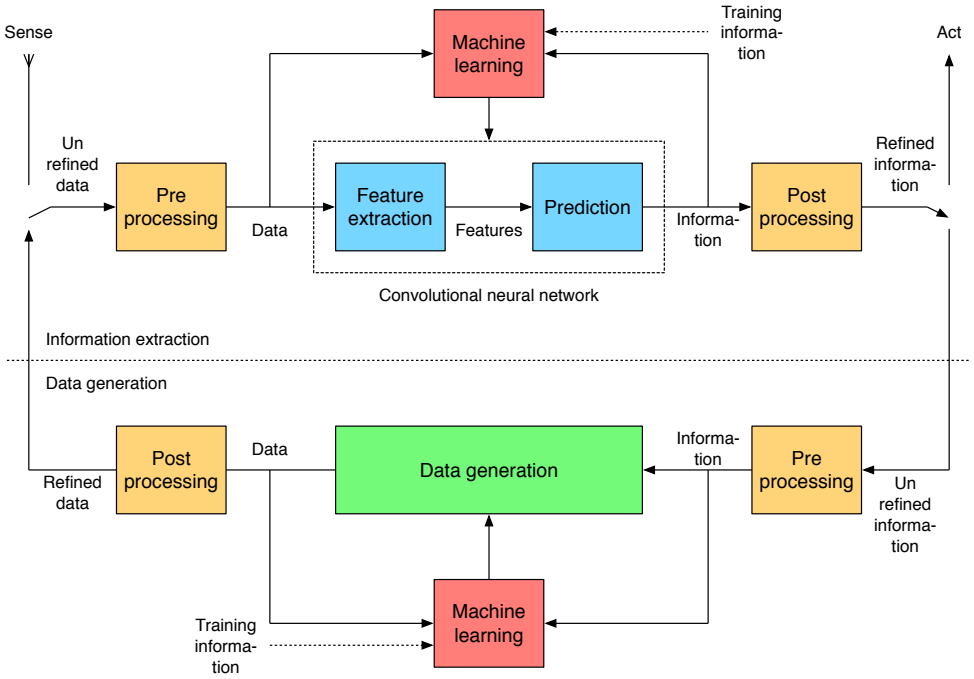
- Reminder
  - Introduction lecture mentioned that there are 2 types of problems
  - Data to information (basically everything we've talked about so far)
  - Information to data (relevant here among other places)

- Thinking
  - Labeling data is miserable
  - So instead invert the problem
    - Which may or may not be easier
  - Synthetic data: start from a label and use an algorithm to generate associated data
  - The data / label combination can then be used for supervised training



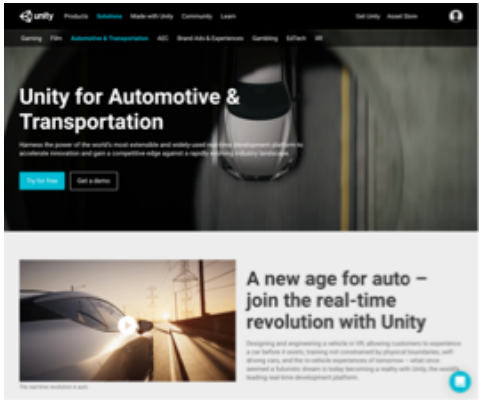
2 Ways To Generate Synthetic Data

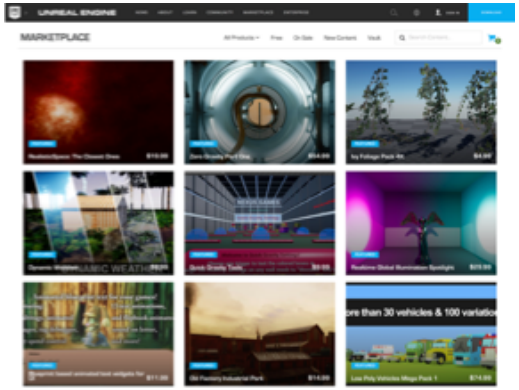
- Reminder
  - When we talked about methods for going from data to information there were 2 basic categories
    - Hand engineered
    - Learned
  - The same applies to going the other direction from information to data
- Hand engineered
  - Requires intelligence on our part
  - Ex: computer vision (thank you to the video game and movie special effects industries)
- Learned
  - Extracts knowledge from nature's information to data generation process to train an algorithm to generate data from information



Hand Engineered

- The information content of the generated synthetic data is limited by the information content of the hand engineered model used to generate the data (but for some cases this is a lot)





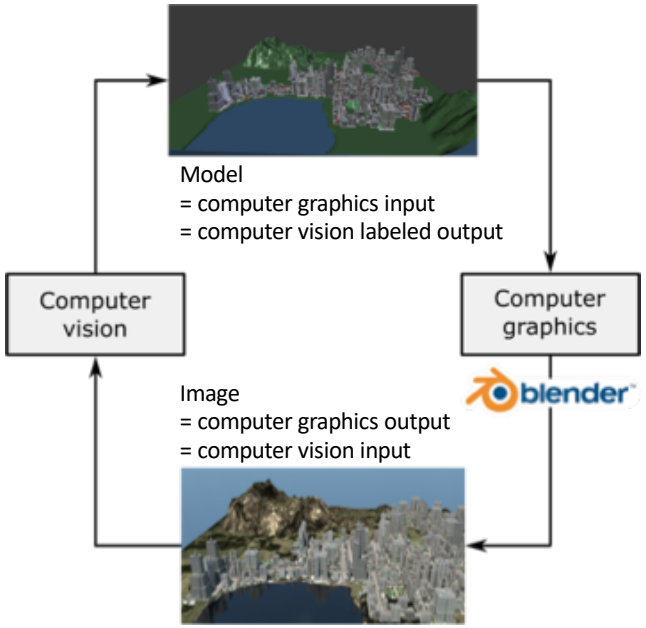
- Examples
  - Computer graphics

Figures from Unreal Engine https://www.unrealengine.com/ and Unity https://unity3d.com

Computer Graphics

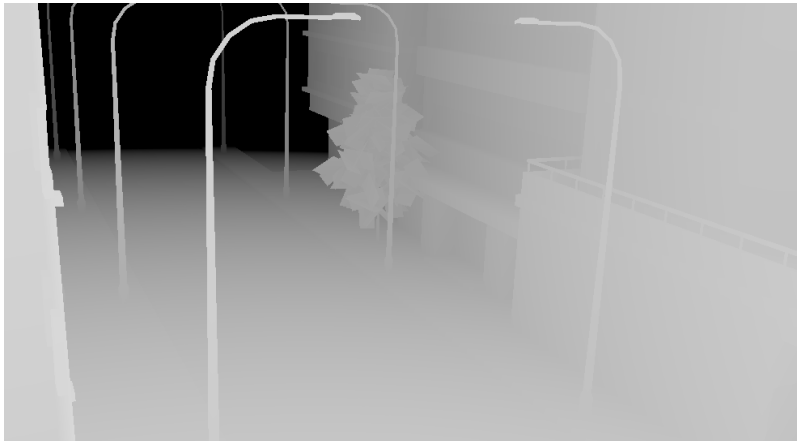
- Computer graphics and computer vision are the opposites (cousins?) of each other
  - We can take advantage of advances in computer graphics from gaming, ... to generate realistic synthetic data

- Example uses
  - RGB images with lenses (industry standard, fisheye warped, ...) and orientations (surround, stereo, ...)
  - Depth and motion
  - Semantic and instance segmentation
  - Events in different conditions and unlikely events



Synthetic Depth Data

Rendered image



Labeled depth

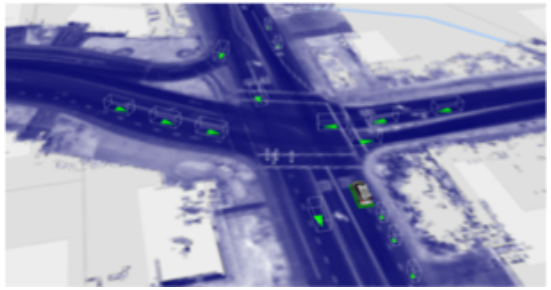
Example RGB and depth renderings
Dense depth labels are difficult to obtain in practice (e.g., KITTI is ~30% labeled)

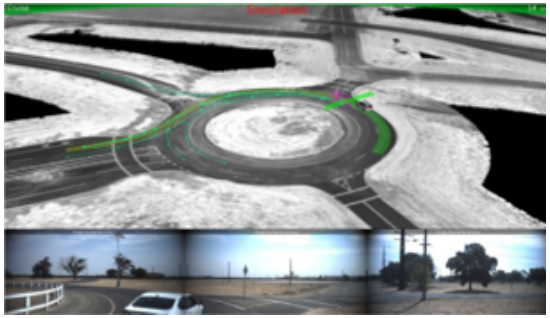
Synthetic Virtual KITTI Data

Virtual worlds as a proxy for multiple-object tracking analysis (https://arxiv.org/abs/1605.06457)

Synthetic Carcraft Data

- Google / Waymo Carcraft
  - > 8 million virtual miles driven per day
  - > 2.5 billion virtual miles diven total





Images from https://www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation-facilities/537648/

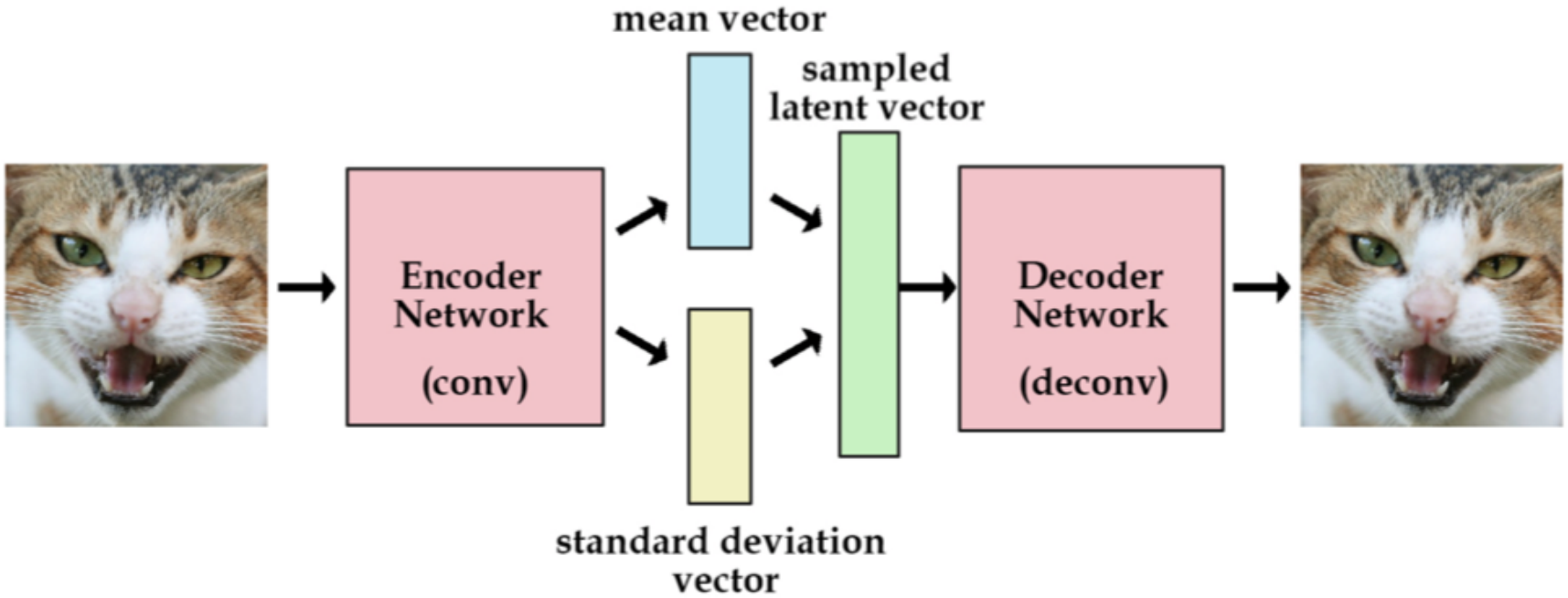
Learned

- Thought chain
  - Neural networks provide a structure for learning to map from data to information
  - So instead of a hand engineered mapping, is it possible to train a neural network to learn to map from information to data (i.e., the other direction)?
  - Subtlety: The information content of the generated synthetic data is limited by the information content of data used for training the algorithm for generating the data

- An auto encoder is an example structure that learns to recreate it's input after it’s input is pushed through a bottleneck
  - The bottleneck forces a representation that contains the key underlying features of the data
  - Can then start at the bottleneck and generate new data (e.g., see variational auto encoder)

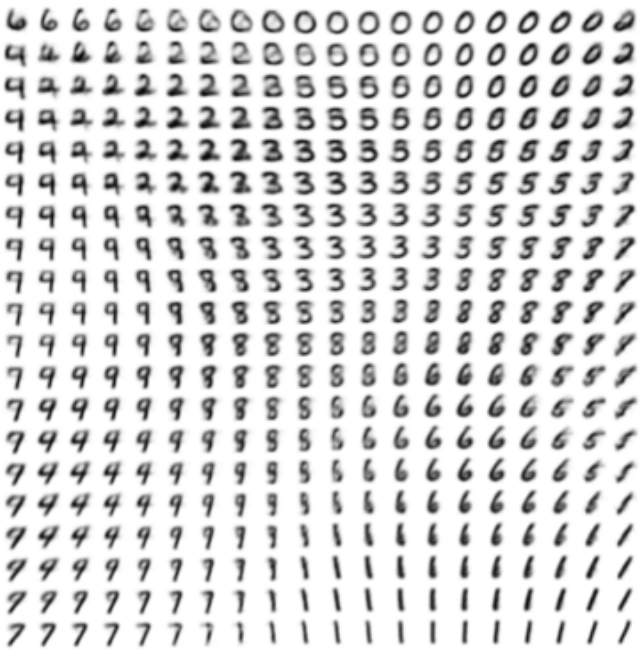
- A generative adversarial network is an example that learns a generative model that generates examples similar to the characteristics of natural samples
  - Typically learned in conjunction with another algorithm for information extraction that determines if data is natural or synthetic

Variational Auto Encoder



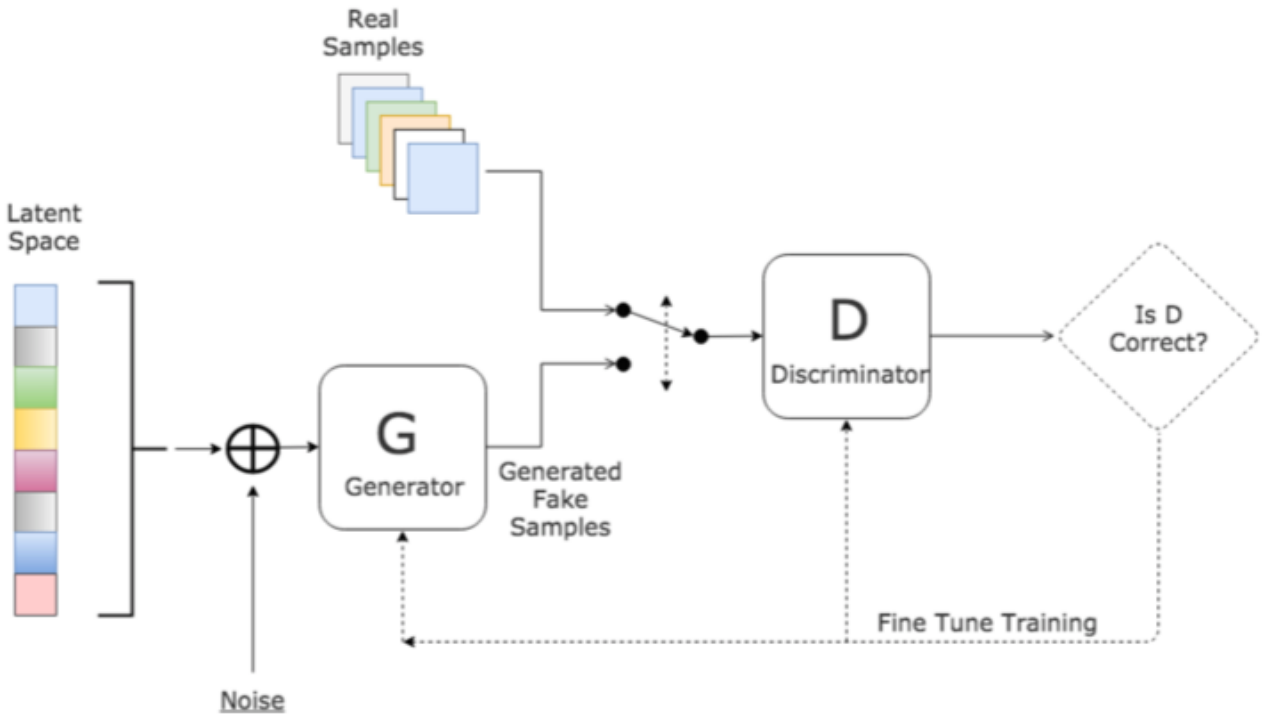
Variational autoencoders explained (http://kvfrans.com/variational-autoencoders-explained/)

Variational Auto Encoder



Auto-encoding variational bayes (https://arxiv.org/abs/1312.6114)

Generative Adversarial Networks



Note: there's a lot of activity in this area and this topic may get it's own focused lecture later in the semester

Generative adversarial networks (https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html)

Data Augmentation

- Basically always want more data to improve training
  - Generate new data by augmenting existing data via modifications
  - Target places where training data is insufficient for learning to achieve a desired level of accuracy on testing data
- Starting point can be real or synthetic data
  - Augmentation can be abstract
    - Warping, flips, color distortions, additive noise, ...
  - Augmentation can be meaningful
    - Changing time of day or weather
- The amount of information added is based on the augmentation process
  - Even if 0 (e.g., noise or a deterministic invertible mapping) it's still potentially useful to training if it helps regularize the learning process

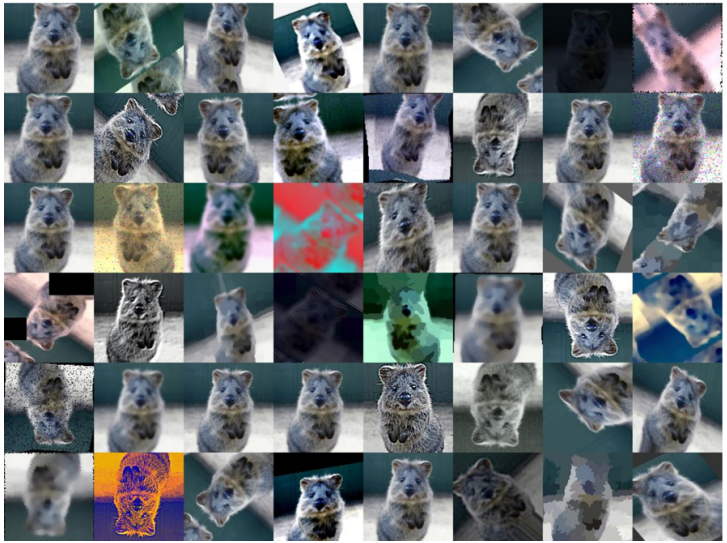


Figure from https://towardsdatascience.com/image-augmentation-for-deep-learning-using-keras-and-histogram-equalization-9329f6ae5085

Synthetic Virtual KITTI Data

Clone

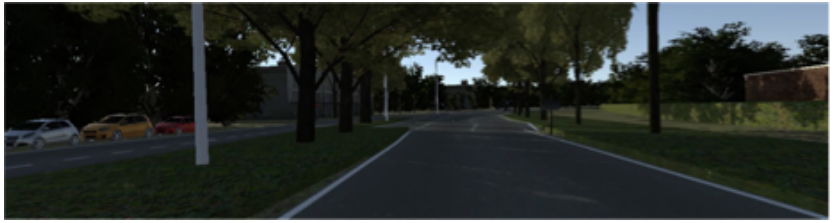
Overcast

Fog

Rain

Morning

Sunset



Adversarial Examples

- High capacity networks that map from data to 1 of many classes can be easily fooled
  - An input that humans are confident in correctly classifying that looks like a typical member of a class
  - But a network is confident in incorrectly classifying to a totally different class
- These types of inputs are called adversarial
- Why create adversarial examples?
  - The good: to use as a data augmentation method for generating additional training samples to make a network more robust
  - The bad: to fool a network into making an incorrect decision for a negative purpose

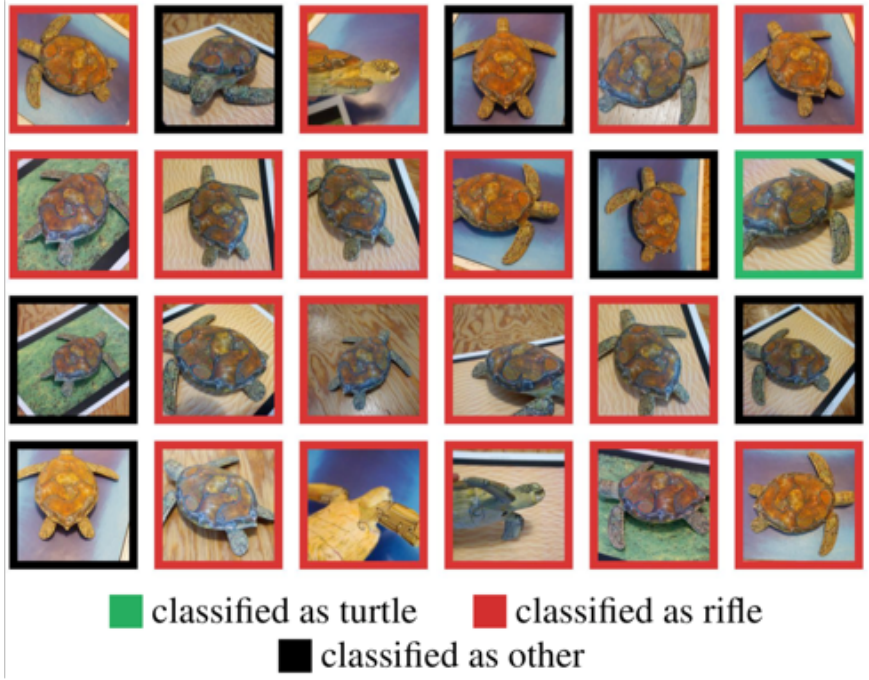


Figure from https://arxiv.org/abs/1707.07397

Generating Adversarial Examples

- Goal: generate examples with the following characteristics
  - Human strongly believes generated data belongs to class A
  - Information extraction algorithm strongly believes generated data belongs to class B
- Strategy
  - Input natural image from class A
  - Input information extraction algorithm
  - Input human perception sensitivity information
  - Use optimization algorithm to compute perturbation of natural image from class A based on the information extraction algorithm and human perception sensitivity that achieves the goals

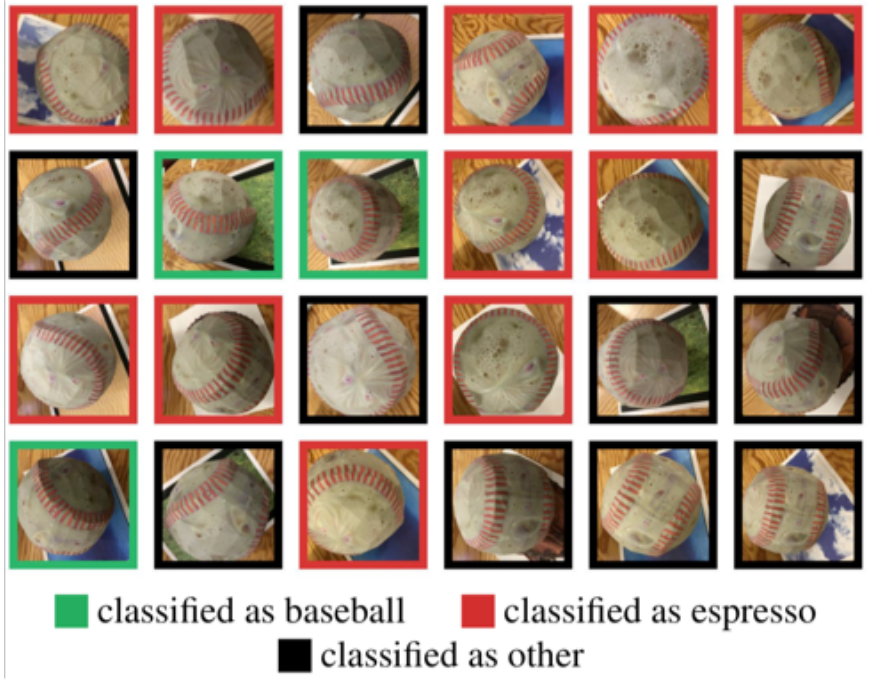


Figure from https://arxiv.org/abs/1707.07397

Selecting Samples Of Training Data

- Batch / iteration / epoch
  - Batch size: a group of inputs (a batch) used to make a single weight update
  - Number of iterations: number of weight updates
  - Number of epochs: batch size * number of iterations / number of training samples

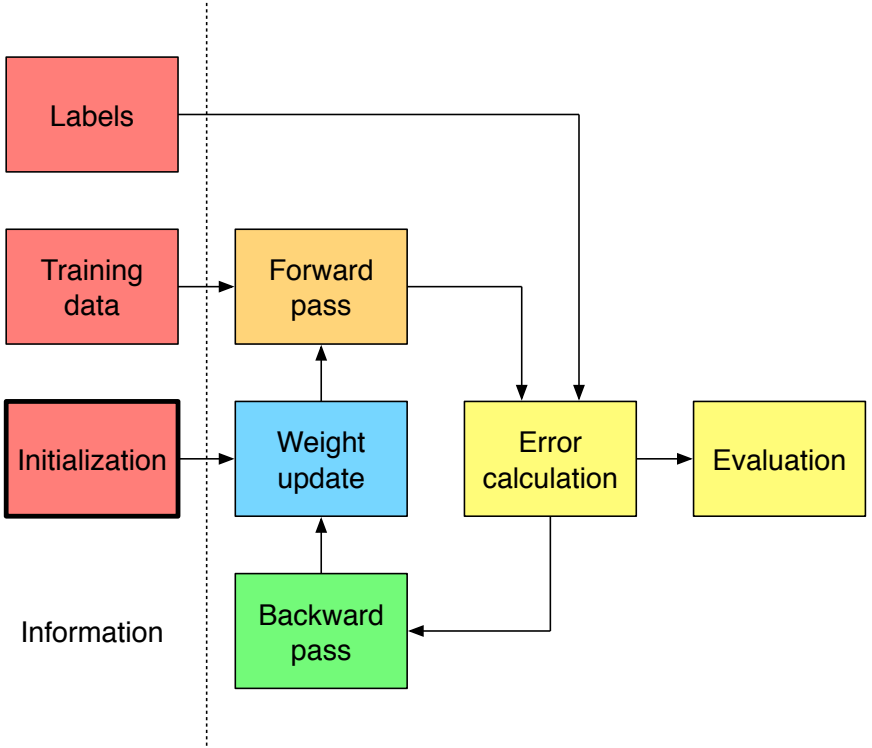
- Options for selecting training samples
  - Walk through a random ordering of all training data a batch at a time
  - Start with easier samples then move to harder samples (curriculum learning)
  - Force some level of balance in different samples
  - Force some bias in different samples
  - More difficult cases vs easy cases (online hard example mining)
  - For subsequent epochs can keep the same ordering or switch to a new random ordering (sometimes a hassle due to practical data preparation constraints)
  - ...

Batch size will be discussed more during the weight update section

Initialization

Strategies

- Random initialization
  - When you only know things probabilistically
- Transfer learning
  - From a different problem
  - From a simpler version of the same problem (curriculum learning)



Random Initialization

- Training a model “from scratch”
- Typically use random initialization
  - Don’t choose all zeros
  - Do use multiple probably independent realizations of a random variable
  - Maybe handle weights different than biases
- Need to determine
  - Distribution type
  - Mean, variance, ...

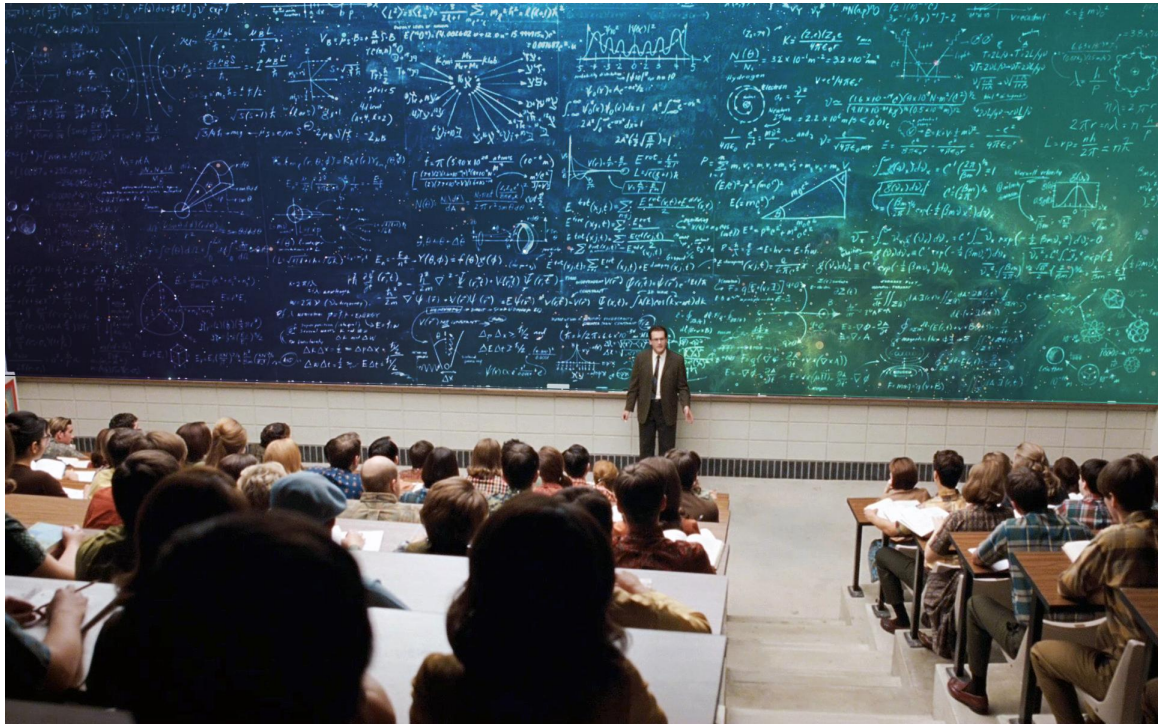


Figure from http://radscreens.com/i/9709/

How To Compute $2^{1/2}$ In 5th Grade

- Algorithm
  - $x = 2^{1/2}$
  - $x^2 = 2$
  - $0 = x^2 - 2$
  - $0 = -\alpha(x^2 - 2)$
  - $X \leftarrow x - \alpha(x^2 - 2)$

- Comment
  - Ideally, you'd like to choose an initialization that's close and in a contractive basin around the true value
  - Practically, when in a million dimensions with a non convex cost function, you're happy to choose a starting point that doesn't lead to divergence

Let $\alpha = 0.1$ and $x_0 = 1.50$

$x_1 = 1.48$
$x_2 = 1.46$
$x_3 = 1.45$
$x_4 = 1.44$
...

Let $\alpha = 0.1$ and $x_0 = 15$

$x_1 = -7$
$x_2 = -12$
$x_3 = -28$
$x_4 = -104$
...

Info Theory And Distribution Choice

- What does selecting a Gaussian distribution mean from an information theoretic perspective?
  - Max entropy distribution when only the mean and variance are known
- What does selecting a uniform distribution mean from an information theoretic perspective?
  - Max entropy distribution when only the domain of support is known
- Question: which is the correct / best choice?
  - What distribution do trained models tend to follow?

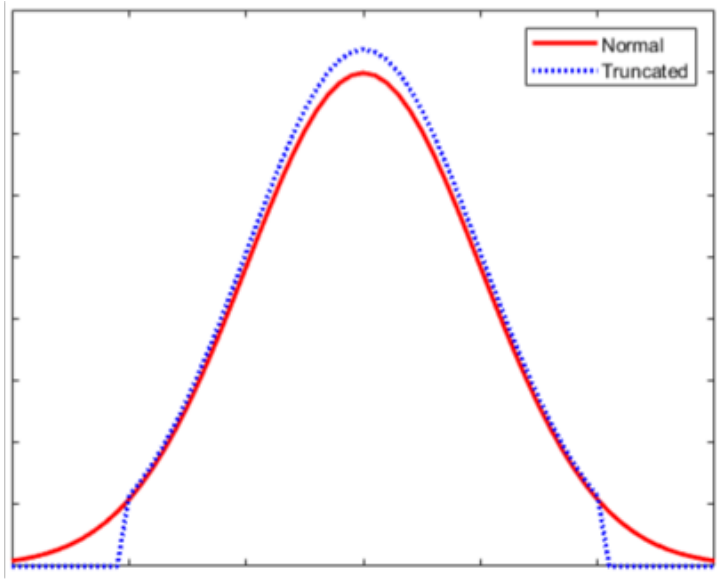


Figure from https://www.mathworks.com/help/stats/prob.normaldistribution.truncate.html

Mean = 0, Standard Deviation = ?

- Understanding the difficulty of training deep feedforward neural networks
  - http://proceedings.mlr.press/v9/glorot10a.html
  - Derived a distribution referred based on an assumption of linear activations and no exploding or vanishing data
  - Glorot / Xavier uniform: [-limit, limit] where limit = sqrt(6/(fan_in + fan_out)) and fan_in and fan_out are the number of input and output units in the weight tensor (paper is for NN, for CNN unclear if fan_in = $N_i*F_r*F_c$ or N_i and fan_out = 1 or N_o)
  - Glorot / Xavier Gaussian: 0 mean and stddev = sqrt(2/(fan_in + fan_out))

- Delving deep into rectifiers: surpassing human-level performance on ImageNet classification
  - https://arxiv.org/abs/1502.01852
  - Follow a similar derivation strategy as Xavier but take the nonlinearity into account and target keeping magnitudes of inputs constant (no exponential growth or shrink of signals)
  - He uniform: [-limit, limit] where limit = sqrt(6/(Fr*Fc*Ni))
  - He Gaussian: 0 mean and standard deviation sqrt(2/(Fr*Fc*Ni)) and biases initialized to 0 (per paper)

Transfer Learning

- Strategy
  - Train the network on a related problem that typically has a lot of data
  - Use those trained parameters as the starting point for the network applied to the problem of interest which typically has less data
  - May need to modify the network head to account for the differences in the problem

- How transferable are features in deep neural networks?
  - https://arxiv.org/pdf/1411.1792.pdf

- Can also use a smaller network to initialize a larger network or a larger network to initialize a smaller network
  - Very deep convolutional networks for large-scale image recognition
    - https://arxiv.org/abs/1409.1556
  - Used a strategy to train a smaller model to initialize a larger model
  - While it helps in some cases, it requires extra training and potentially has issues with getting stuck in sub optimal local minima

How This Impacts You This Semester

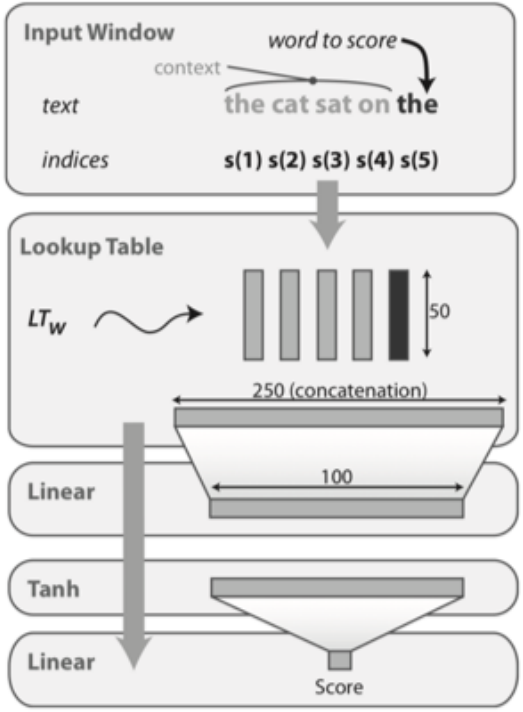
- Training a ~ large model like ResNet 50 on ImageNet using a fast gaming system with 2x Nvidia GTX 1080 Ti GPUs can easily take a couple of weeks to a month
  - And it's very easy to mess up and have to experiment a few times with hyper parameters to get it to work
- Transfer learning let's you take advantage of the work someone else did training the network to give you a jump start as you apply it to a different but related enough problem

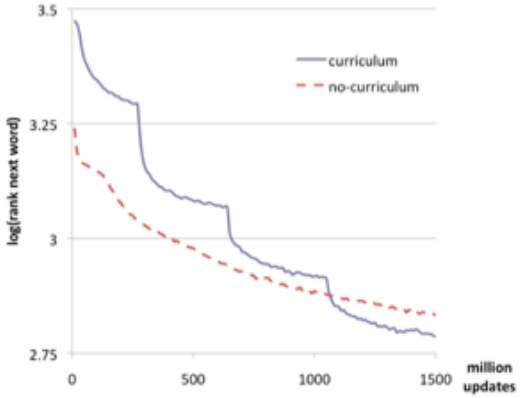
Figures from https://github.com/tensorflow/models/tree/master/official and https://pytorch.org/docs/stable/torchvision/models.html

Curriculum Learning

- This sits 1/2 way between the data and initialization sections

- Strategy
  - Mimic how humans learn and orders training samples typically from easier to more difficult
  - Potentially improves accuracy and generalization
  - So it's a data selection method
  - But training on easier samples can be thought of as an initialization for subsequent training on more complex samples



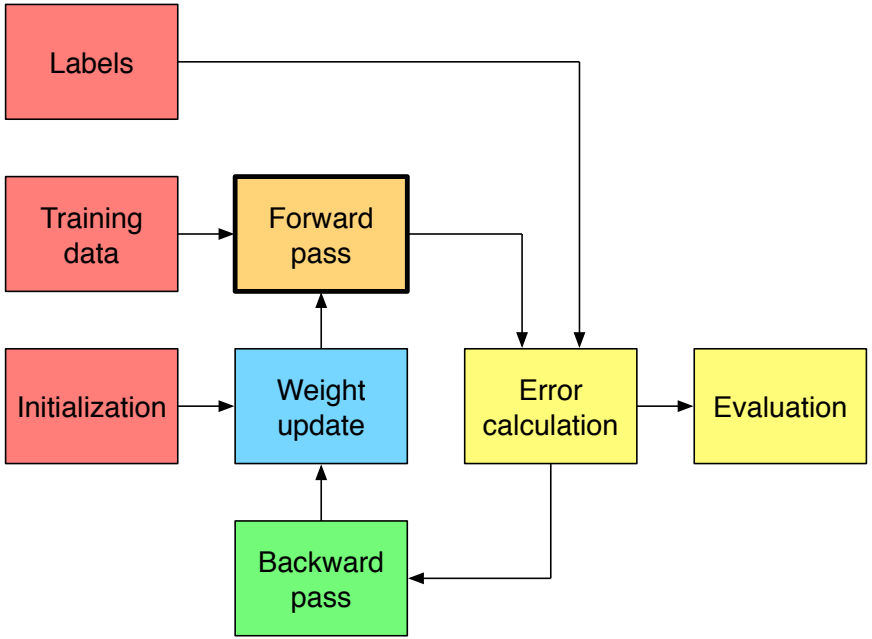


Curriculum learning (https://ronan.collobert.com/pub/matos/2009_curriculum_icml.pdf)

Forward Pass

CNN Training Vs Function Optimization

- Modifications to the forward pass for 2 reasons
  - Convergence: improve parameter estimation on training data
  - Regularization: improve performance on testing data



Convergence

- Strategy
  - Modify the network structure (forward pass) to improve convergence during training (backward pass)
  - After training, the network structure modification can frequently be absorbed into the original network
- Examples
  - Batch normalization
  - Batch renormalization
  - Group normalization

Batch Normalization

- Notes
  - Initialization played all sorts of games to prevent exploding or vanishing signals as they propagate through the network
  - Idea: why not add a data dependent layer after convolution that on a per channel basis normalizes the data to 0 mean unit variance?
  - The benefits of this were described in the paper Efficient backprop (http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf)
  - Note that the addition of this layer does not increase the set of functions the network can approximate in the forward path, it only improves training

- Batch normalization
  - For each batch compute a per channel mean and variance
  - Normalize each channel to ~ 0 mean ~ 1 variance
  - Includes learnable scale and shift parameters to maintain expressiveness in transformation
  - If possible absorb the scale and shift parameters into a neighboring layer during testing

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figure from original paper

Batch Renormalization

- Issues with batch normalization
  - Different operations during training and testing
  - If small batch size then large changes between batches

- Batch renormalization
  - Start with batch norm
  - Start accumulating running averages for the mean and variance
  - Then gradually transition from using the per sample mean and variance to the running average mean and variance

Input: Values of x over a training mini-batch $\mathcal{B} = \{x_{1...m}\}$; parameters γ, β; current moving mean μ and standard deviation σ; moving average update rate α; maximum allowed correction $r_{\max}, d_{\max}$.
Output: $\{y_i = \text{BatchRenorm}(x_i)\}$; updated μ, σ.

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_{\mathcal{B}} \leftarrow \sqrt{\epsilon + \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2}$$

$$r \leftarrow \texttt{stop\_gradient}\left(\text{clip}_{[1/r_{\max}, r_{\max}]}\left(\frac{\sigma_{\mathcal{B}}}{\sigma}\right)\right)$$

$$d \leftarrow \texttt{stop\_gradient}\left(\text{clip}_{[-d_{\max}, d_{\max}]}\left(\frac{\mu_{\mathcal{B}} - \mu}{\sigma}\right)\right)$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}} \cdot r + d$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta$$

$$\mu := \mu + \alpha(\mu_{\mathcal{B}} - \mu) \qquad \text{// Update moving averages}$$

$$\sigma := \sigma + \alpha(\sigma_{\mathcal{B}} - \sigma)$$

Inference: $y \leftarrow \gamma \cdot \frac{x - \mu}{\sigma} + \beta$

Figure from original paper

Group Normalization

- Issues with batch normalization
  - Different operations during training and testing
  - If small batch size then large changes between batches
- Group normalization
  - Divide channels into groups
  - Compute mean and variance based on groups of channels

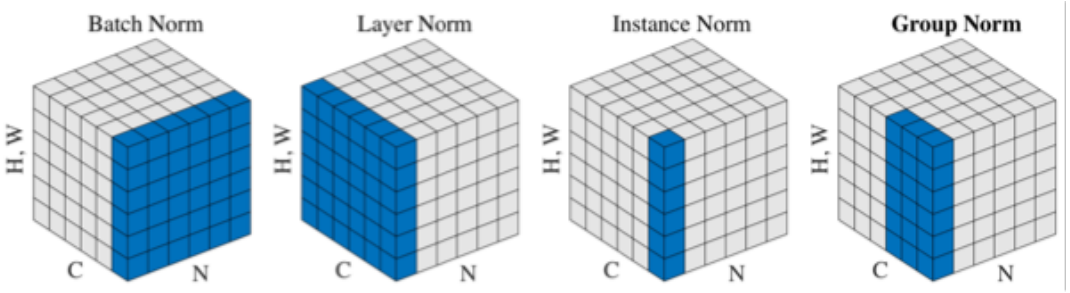
```
def GroupNorm(x, gamma, beta, G, eps=1e−5):
    # x: input features with shape [N,C,H,W]
    # gamma, beta: scale and offset, with shape [1,C,1,1]
    # G: number of groups for GN

    N, C, H, W = x.shape
    x = tf.reshape(x, [N, G, C // G, H, W])

    mean, var = tf.nn.moments(x, [2, 3, 4], keep_dims=True)
    x = (x − mean) / tf.sqrt(var + eps)

    x = tf.reshape(x, [N, C, H, W])

    return x * gamma + beta
```



C = channel dimension
N = batch dimension
H = feature map height
W = feature map width

Figures from original paper

Regularization

- It’s possible to modify the forward path to help with regularization
- Example network modifications to improve regularization
  - Stochastic width
  - Stochastic depth
  - Noise addition in the network

Stochastic Width (Dropout, DropConnect)

- The ideas behind stochastic width were most helpful in improving training with (multiple) large fully connected layers in earlier CNN designs
  - For wide networks this may be desirable
  - For narrow networks where the number of classes is on the order of or greater than the number of features this may be undesirable
  - Common current designs with global average pool and a single relatively smaller fully connected layer typically don't need to use stochastic width; but regardless, it's still a useful technique to know

- Dropout
  - Dropout zeros out a random set of layer outputs per batch
  - Implicitly it forces multiple groups of output features to be able to estimate a class

- DropConnect
  - DropConnect zeros out a random set of layer weights per batch
  - Implicitly it forces multiple groups of input features to be able to generate and output feature

Stochastic Width (Dropout, DropConnect)

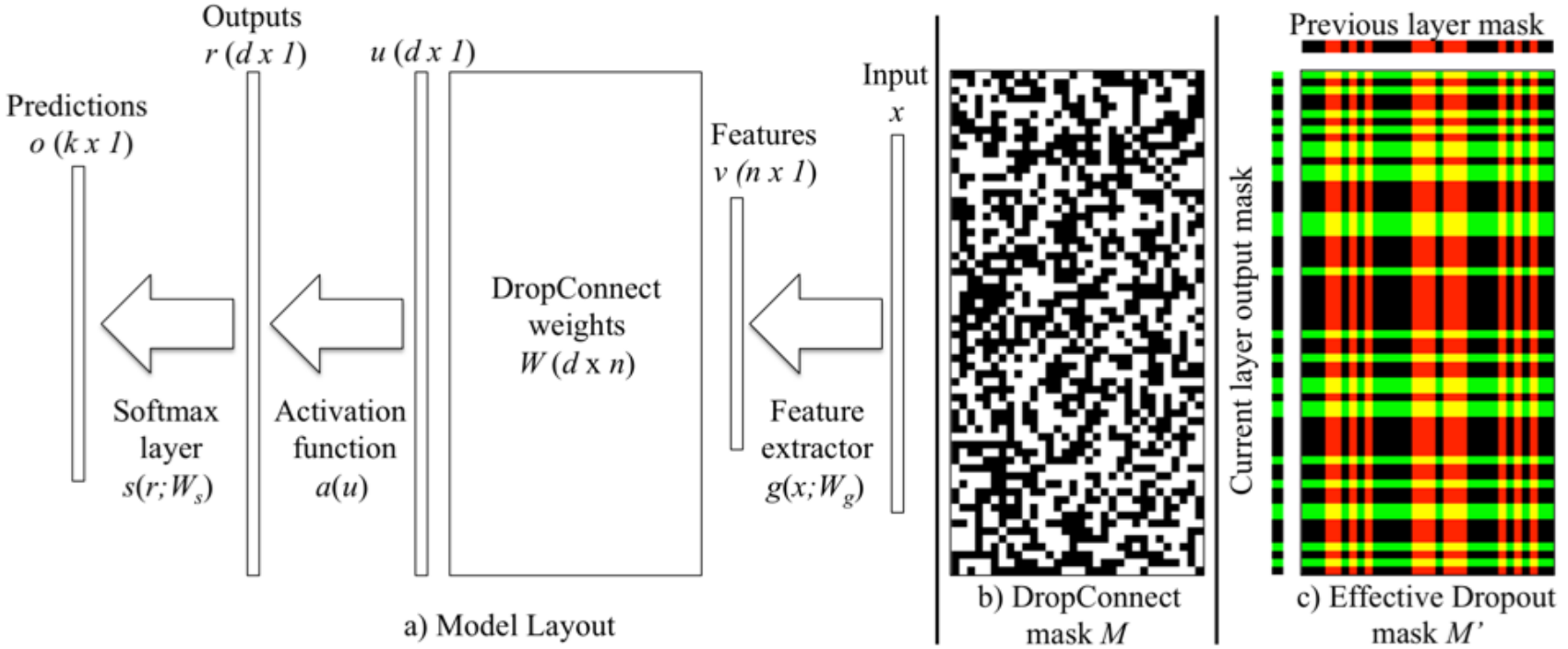
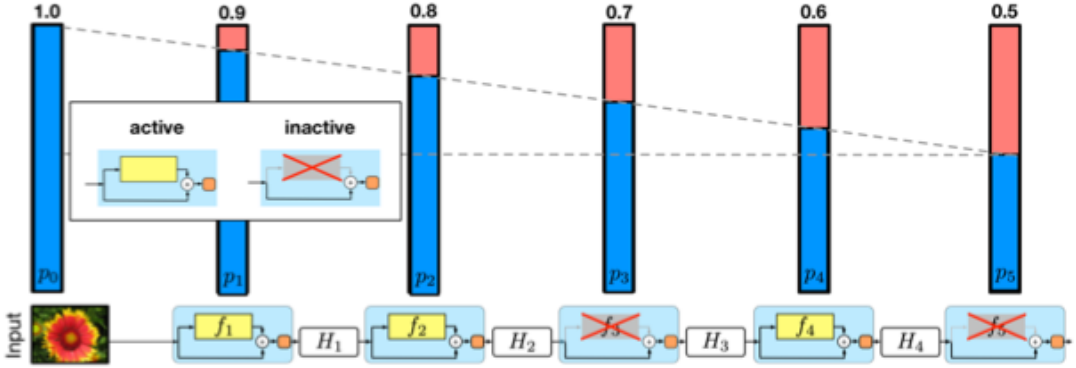


Figure from https://cs.nyu.edu/~wanli/dropc/dropc.pdf

Stochastic Depth (Layer Skipping)

- Stochastic depth (layer / building block skipping)
  - Only really used in very very deep residual networks
  - Implicitly it forces layers to do iterative refinement

- Notes
  - Probability of a layer being on or skipped (identity) is recommended to be layer dependent
  - Earlier in the network a layer is more likely to be on
  - Later in the network a layer is more likely to be bypassed
  - In testing a network output is scaled based on the probability that it was on during training



Note that H_n refers to refers to the feature map at the output of operation n, not the layer itself

Figure from https://arxiv.org/abs/1603.09382

Noise Addition

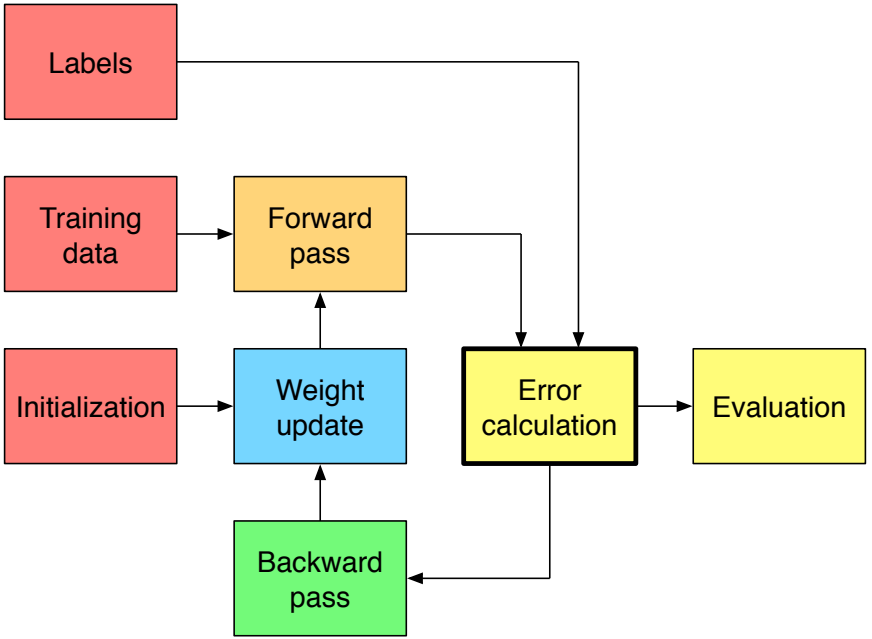
- Noise addition in the network
  - Feature maps
  - ReLU 0 point
  - ...

- Examples
  - Estimating or propagating gradients through stochastic neurons
    - https://arxiv.org/abs/1305.2982
  - Noisy activation functions
    - https://arxiv.org/abs/1603.00391
  - Dataset augmentation in feature space
    - https://arxiv.org/abs/1702.05538

Error Calculation

CNN Training Vs Function Optimization

- In the case of classification
  - Training with 1 loss (e.g., soft max – cross entropy)
  - Testing with a different loss (e.g., arg max)



2 Broad Categories Of Loss Functions

- Classification
  - Prediction in a finite series of classes
  - Typically ...
    - Easier than regression
    - Complexity increases with number of classes
    - Complexity increases with similarity of classes

- Regression
  - Prediction in a continuum of values
  - Typically ...
    - More difficult than classification
    - Complexity increases with range and required resolution

Classification Loss

- Distinct outputs (classes)
- Softmax cross entropy loss function
  - Soft max effectively converts network outputs to a probability mass function
  - KL divergence for comparing 2 probability mass functions: target and network output
  - For a 1 hot target probability mass function KL divergence reduces to cross entropy
  - The calculus lecture notes derived a nice form for the gradient of the error at the output of soft max cross entropy with the input feature map to soft max
- Other options are possible
  - KL divergence with label smoothing, noise or overconfidence penalization
  - Large margin softmax
  - Optimal transport
  - ...

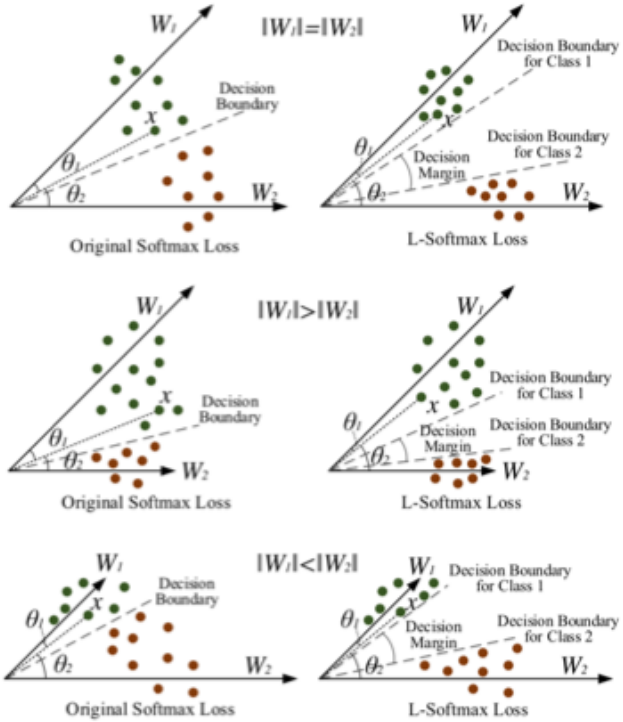


Figure from https://arxiv.org/abs/1612.02295

Regression Loss

- Continuous valued output
  - Strategy: if you can quantize the continuous range and use classification instead of regression it may be better
- Standard l_p norms
  - p = 1
  - p = 2
- Other options are possible
  - Huber loss / smooth L1 loss (like l_2 for small values and l_1 for large values)

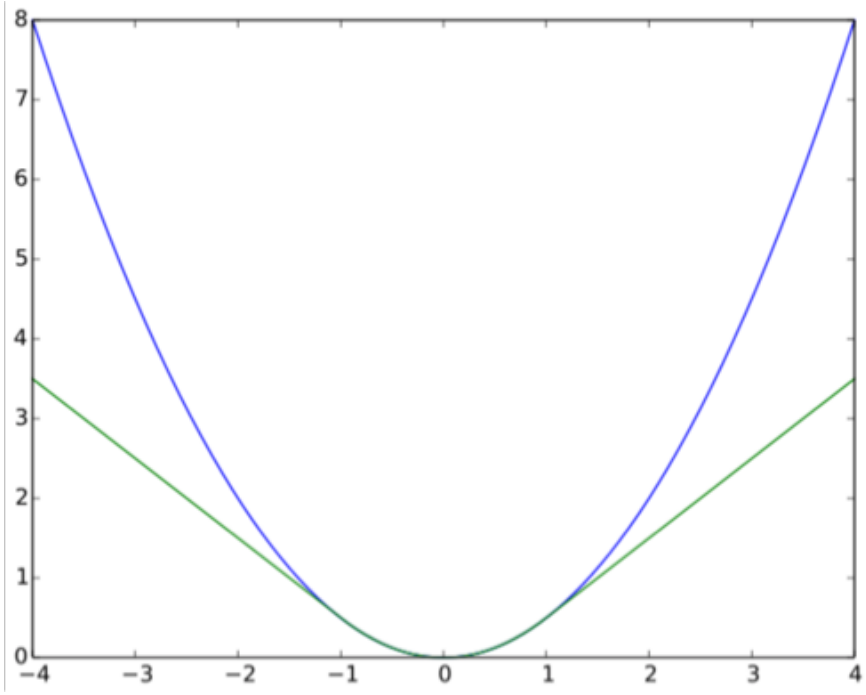
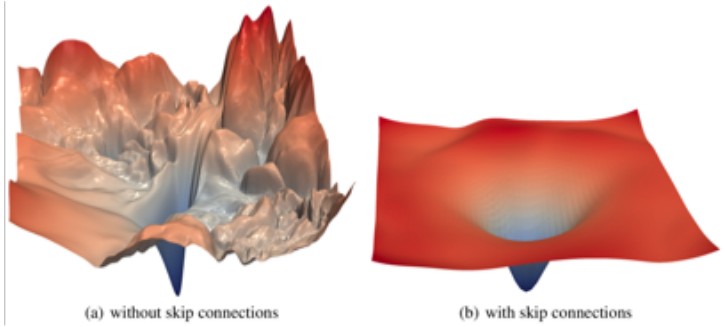


Figure from https://en.wikipedia.org/wiki/Huber_loss#/media/File:Huber_loss.svg

Loss Surface Shapes

- The shape is a consequence of the choice of the data, network design and loss function
  - The weight update is going to attempt to find the lowest value of the function (and assume it also corresponds to the optimal value on the testing data)
  - Unfortunately, the loss surface is not convex
- You can't visualize it
  - You can think of shapes in ~ 1 – 4 dimensions
  - The error (loss) is a function of the number of parameters (easily millions)
  - You can't (easily) think of shapes in millions dimensions
- But it has some characteristics that allow training to work
- Reminder: tell story on how to escape from jail



Figures from https://arxiv.org/abs/1712.09913

Unequal Class Weightings

- Some classes are more difficult than others
- But so far the error calculation has treated all classes the same
- Strategy for modifications to the equal output class weighting of errors
  - Weight classes that are difficult (likely under represented in the training data) for the network more in the error computation
- Focal loss for dense object detection
  - https://arxiv.org/abs/1708.02002
  - Modifies cross entropy based on the inverse of category likelihood and probability when misclassified

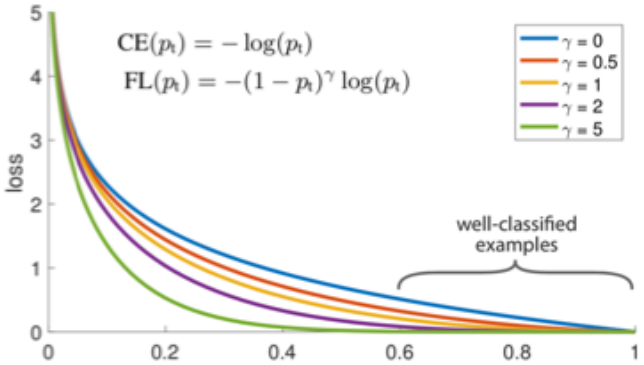
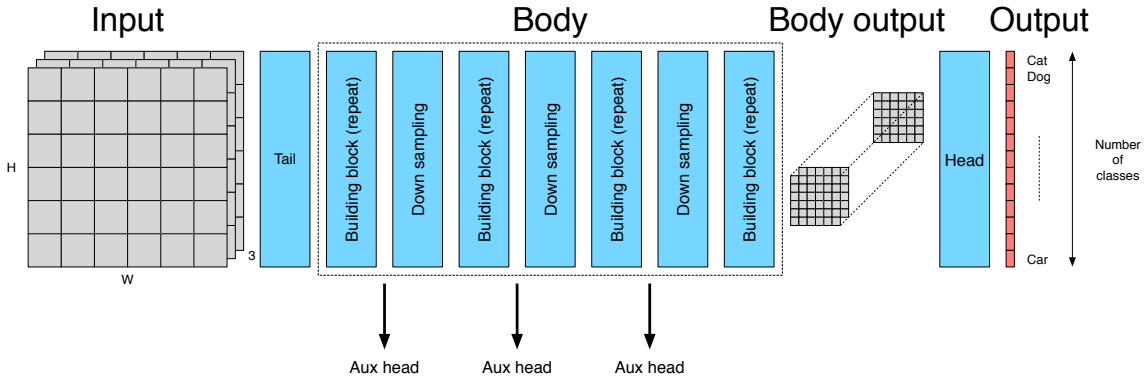


Figure from https://arxiv.org/abs/1708.02002

Auxiliary Network Heads

- Features strong but poorly localized at the output of the body
  - Features are better localized but weaker earlier on in the network
  - While weaker, they're potentially still strong enough to predict classes, just not as accurately
- Idea: use auxiliary network heads during training to provide gradient information directly into the middle of the network
- Note: has possible drawback of getting the parameters stuck too much at a local minima



Multiple Network Heads

- Sometimes a network tail and body have multiple heads, each that accomplishes a different task
  - Want to optimize the performance of the full network
  - So need to create a loss that's combines the losses of all the heads
  - Multi task / combined function

- It's common to do this in multiple object detection networks
  - Classification of boxes as bounding objects and dimension modifications
  - Classification of objects in the bounding boxes

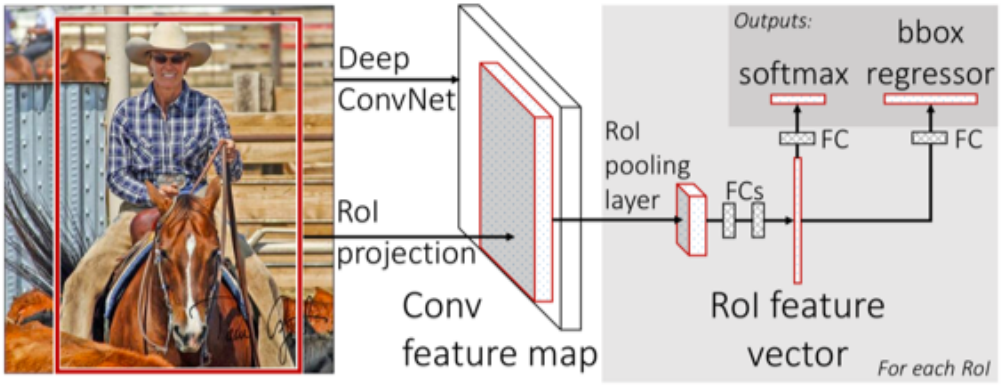
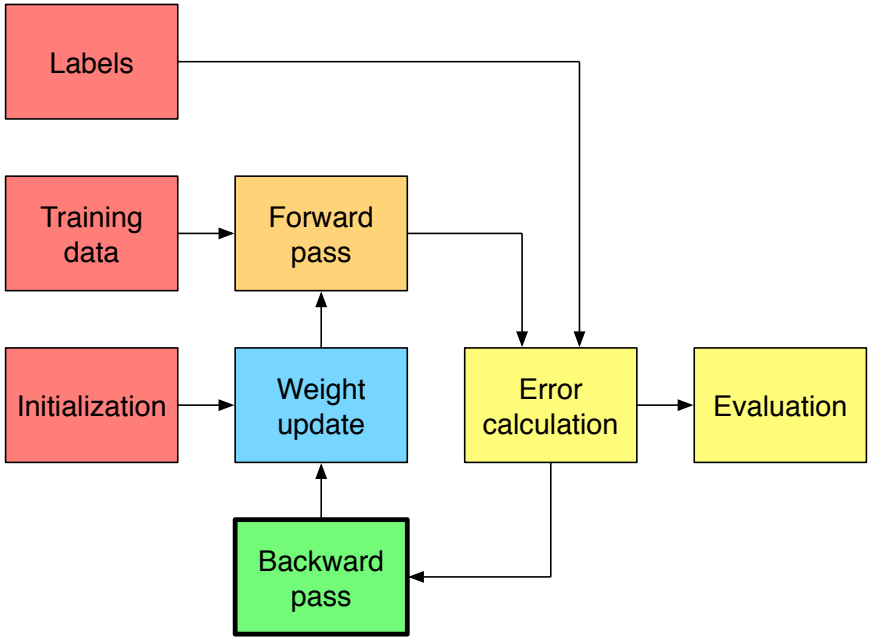


Figure from https://arxiv.org/abs/1504.08083

Backward Pass

Running Out Of Memory Is No Fun

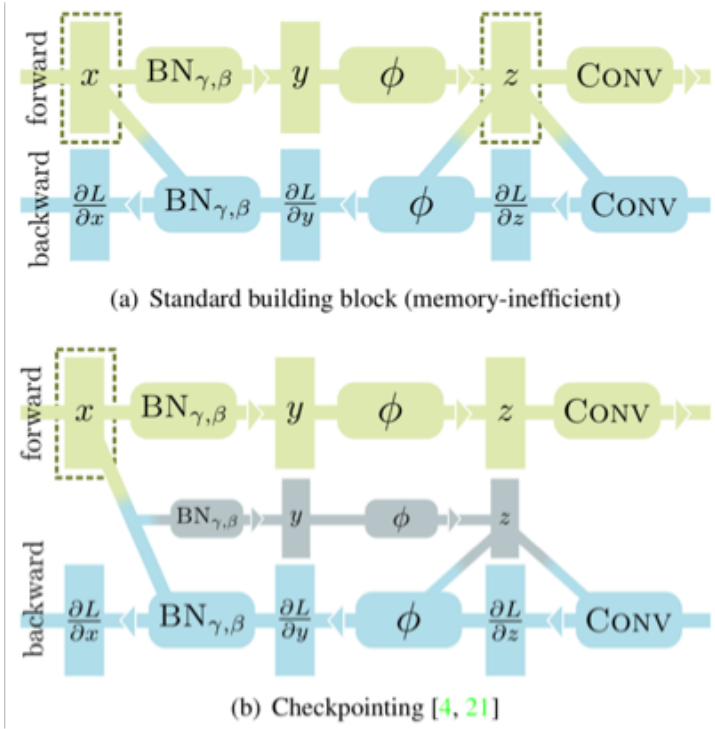
- Reverse mode automatic differentiation
  - Covered in the calculus lecture
  - Basically stays the same
- But possibly make a modifications to address a practical issue: memory (or a lack thereof)
- Situation
  - Typically want larger batch sizes
  - This increases memory
  - Note that gradient computations in the backward pass are dependent on feature maps in the forward pass
  - End up running out of memory
  - Also, moving memory around is typically less efficient then computing



Checkpointing

- A strategy for addressing running out of memory
  - Trade increased computation for reduced memory
  - A collection of techniques that falls under the umbrella of checkpointing
  - Also, change some non invertible operations in the forward pass to similar but invertible versions and use these in a similar way

- The data-flow equations of checkpointing in reverse automatic differentiation
  - https://www-sop.inria.fr/tropics/papers/DauvergneHascoet06.pdf
- Training deep nets with sublinear memory cost
  - https://arxiv.org/pdf/1604.06174.pdf
- Memory-efficient backpropagation through time
  - https://arxiv.org/abs/1606.03401
- For a nice implementation of this and figures see https://github.com/openai/gradient-checkpointing



(a) Standard building block (memory-inefficient)

(b) Checkpointing [4, 21]

Figure from https://arxiv.org/abs/1712.02616

In Place Activated Batch Norm

- Strategy
  - Replaces batch norm -> activation with a single layer and uses leaky ReLU vs ReLU to allow inversion through the nonlinearity to save ~ 50% of memory
  - Remember the word “bijection” from the discussion of functions

- In-place activated batchnorm for memory-optimized training of DNNs
  - https://arxiv.org/abs/1712.02616
  - See also: https://blog.mapillary.com/update/2017/12/08/massive-memory-savings-for-training-modern-deep-learning-architectures.html

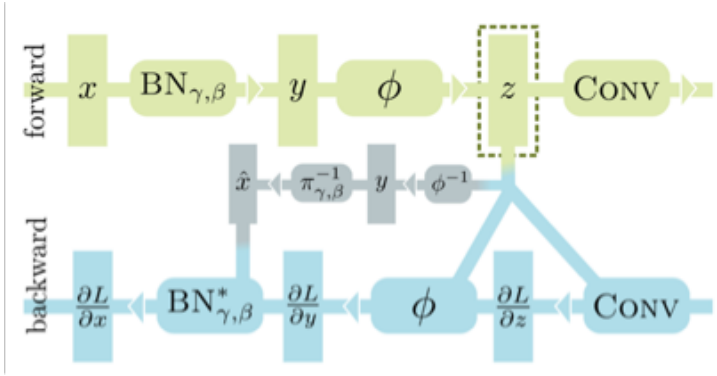


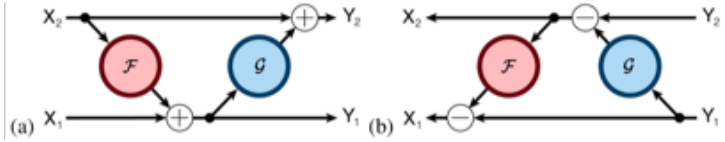
Figure from https://arxiv.org/abs/1712.02616

Reversible Architectures

- Strategy
  - In place activated batch norm made the pointwise nonlinearity invertible
  - Question: is it possible to make the whole network (or at least large blocks of it) invertible

- RevNet
  - Apologies: I know you thought you were done learning new xNet names
  - Defines a different residual network such that each layers output can be derived from the previous layers output
  - The reversible residual network: backpropagation without storing activations
    - https://arxiv.org/pdf/1707.04585.pdf

- Perhaps less popular to do this in practice, but it's worth being aware of as a generally useful idea



Technique	Spatial Complexity (Activations)	Computational Complexity
Naive	$\mathcal{O}(L)$	$\mathcal{O}(L)$
Checkpointing [20]	$\mathcal{O}(\sqrt{L})$	$\mathcal{O}(L)$
Recursive Checkpointing [5]	$\mathcal{O}(\log L)$	$\mathcal{O}(L \log L)$
Reversible Networks (Ours)	$\mathcal{O}(1)$	$\mathcal{O}(L)$

Figures from https://arxiv.org/pdf/1707.04585.pdf

References

Generalization

- Regularization for deep learning: a taxonomy
  - https://arxiv.org/abs/1710.10686
- Generalization in deep learning
  - https://arxiv.org/abs/1710.05468
- Deep learning and generalization
  - https://www.lpsm.paris/conf_lpsm/SlideBousquetParis2018.pdf

Data

- ImageNet: a large-scale hierarchical image database
  - https://ieeexplore.ieee.org/document/5206848
- ImageNet large scale visual recognition challenge
  - https://arxiv.org/abs/1409.0575
- The PASCAL visual object classes (VOC) challenge
  - http://host.robots.ox.ac.uk/pascal/VOC/pubs/everingham10.pdf
- Microsoft COCO
  - http://cocodataset.org/
- The Cityscapes dataset for semantic urban scene understanding
  - https://arxiv.org/abs/1604.01685
  - https://www.cityscapes-dataset.com

Data

- Amazon mechanical turk
  - https://www.mturk.com
- VoTT: visual object tagging tool
  - https://github.com/Microsoft/VoTT
- VGG image annotator (VIA)
  - http://www.robots.ox.ac.uk/~vgg/software/via/
- ALP's label tool
  - https://alpslabel.wordpress.com
- LabelMe
  - http://labelme2.csail.mit.edu/Release3.0/index.php

- A closer look at memorization in deep networks
  - https://arxiv.org/abs/1706.05394
- Virtual worlds as a proxy for multiple-object tracking analysis
  - https://arxiv.org/abs/1605.06457

Data

- Auto-encoding variational bayes
  - https://arxiv.org/abs/1312.6114
- Generative adversarial networks
  - https://arxiv.org/abs/1406.2661
- Generative adversarial networks (GANs)
  - https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf
- Generative models
  - http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf
- Introduction to GANs
  - http://www.iangoodfellow.com/slides/2018-06-22-gan_tutorial.pdf
- A beginner's guide to generative adversarial networks (GANs)
  - https://skymind.ai/wiki/generative-adversarial-network-gan

- AutoAugment: learning augmentation policies from data
  - https://arxiv.org/abs/1805.09501

Data

- Intriguing properties of neural networks
  - https://arxiv.org/abs/1312.6199
- Deep neural networks are easily fooled: high confidence predictions for unrecognizable images
  - https://arxiv.org/abs/1412.1897
- Explaining and harnessing adversarial examples
  - https://arxiv.org/abs/1412.6572
- Towards deep learning models resistant to adversarial attacks
  - https://arxiv.org/pdf/1706.06083.pdf
- Synthesizing robust adversarial examples
  - https://arxiv.org/abs/1707.07397
- Training region-based object detectors with online hard example mining
  - https://arxiv.org/abs/1604.03540

Initialization

- How transferable are features in deep neural networks?
  - https://arxiv.org/abs/1411.1792
- Curriculum learning
  - https://ronan.collobert.com/pub/matos/2009_curriculum_icml.pdf
- Automated curriculum learning for neural networks
  - https://arxiv.org/abs/1704.03003

Forward Pass

- Batch normalization: accelerating deep network training by reducing internal covariate shift
  - https://arxiv.org/abs/1502.03167
- Batch renormalization: towards reducing minibatch dependence in batch-normalized models
  - https://arxiv.org/abs/1702.03275
- Group normalization
  - https://arxiv.org/abs/1803.08494
- Improving neural networks by preventing co-adaptation of feature detectors
  - https://arxiv.org/abs/1207.0580

- Dropout: a simple way to prevent neural networks from overfitting
  - https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf
- Regularization of neural networks using dropconnect
  - https://cs.nyu.edu/~wanli/dropc/dropc.pdf
- Deep networks with stochastic depth
  - https://arxiv.org/abs/1603.09382

Forward Pass

- Dataset augmentation in feature space
  - https://arxiv.org/abs/1702.05538
- Estimating or propagating gradients through stochastic neurons
  - https://arxiv.org/abs/1305.2982
- Noisy activation functions
  - https://arxiv.org/abs/1603.00391

Error Calculation

- Large-margin softmax loss for convolutional neural networks
  - https://arxiv.org/abs/1612.02295
- Robust estimation of a location parameter
  - https://projecteuclid.org/download/pdf_1/euclid.aoms/1177703732

- Transport-based analysis, modeling, and learning from signal and data distributions
  - https://arxiv.org/abs/1609.04767
- Optimal mass transport: signal processing and machine-learning applications
  - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6024256/
- Applications of optimal transport to machine learning and signal processing
  - https://mathematical-coffees.github.io/slides/mc01-courty.pdf

Error Calculation

- Distilling the knowledge in a neural network
  - https://arxiv.org/abs/1503.02531
- Rethinking the inception architecture for computer vision
  - https://arxiv.org/abs/1512.00567
- DisturbLabel: regularizing CNN on the loss layer
  - https://arxiv.org/abs/1605.00055
- Regularizing neural networks by penalizing confident output distributions
  - https://arxiv.org/abs/1701.06548

Error Calculation

- The loss surfaces of multilayer networks
  - https://arxiv.org/abs/1412.0233
- Identifying and attacking the saddle point problem in high-dimensional non-convex optimization
  - https://arxiv.org/abs/1406.2572
- Visualizing the loss landscape of neural nets
  - https://arxiv.org/abs/1712.09913
- Comparing dynamics: deep neural networks versus glassy systems
  - https://arxiv.org/abs/1803.06969

Error Calculation

- Focal loss for dense object detection
  - https://arxiv.org/abs/1708.02002
- Hierarchical loss for classification
  - https://arxiv.org/abs/1709.01062
- Deeply-supervised nets
  - https://arxiv.org/abs/1409.5185
- Going deeper with convolutions
  - https://arxiv.org/abs/1409.4842

Backward Pass

- Automatic differentiation
  - http://www.robots.ox.ac.uk/~tvg/publications/talks/autodiff.pdf
- Reverse-mode automatic differentiation: a tutorial
  - https://rufflewind.com/2016-12-30/reverse-mode-automatic-differentiation
- Automatic differentiation of algorithms
  - https://www.sciencedirect.com/science/article/pii/S037704270004222?via%3Dihub
- Automatic reverse-mode differentiation: lecture notes
  - http://www.cs.cmu.edu/~wcohen/10-605/notes/autodiff.pdf

Weight Update

- Large-scale machine learning with stochastic gradient descent
  - http://leon.bottou.org/publications/pdf/compstat-2010.pdf
- Why momentum really works
  - https://distill.pub/2017/momentum/
- On the importance of initialization and momentum in deep learning
  - http://www.cs.toronto.edu/~fritz/absps/momentum.pdf
- An overview of gradient descent optimization algorithms
  - https://arxiv.org/abs/1609.04747
- Optimization methods for large-scale machine learning
  - https://arxiv.org/abs/1606.04838
- Online learning rate adaptation with hypergradient descent
  - https://arxiv.org/abs/1703.04782
- ADAM: a method for stochastic optimization
  - https://arxiv.org/pdf/1412.6980.pdf
- On the convergence of ADAM and beyond
  - https://openreview.net/pdf?id=ryQu7f-RZ

Weight Update

- Nesterov accelerated gradient and momentum
  - https://jlmelville.github.io/mize/nesterov.html
- Neural optimizer search with reinforcement learning
  - https://arxiv.org/abs/1709.07417
- The marginal value of adaptive gradient methods in machine learning
  - https://arxiv.org/pdf/1705.08292.pdf
- When is a convolutional filter easy to learn?
  - https://research.fb.com/wp-content/uploads/2018/04/when-is-a-convolutional-filter-easy-to-learn.pdf

- Don't decay the learning rate, increase the batch size
  - https://arxiv.org/abs/1711.00489
- Revisiting small batch training for deep neural networks
  - https://arxiv.org/abs/1804.07612
- Train longer, generalize better: closing the generalization gap in large batch training of neural networks
  - https://arxiv.org/abs/1705.08741

Weight Update

- Sensitivity and generalization in neural networks: an empirical study
  - https://arxiv.org/abs/1802.08760
- Robust large margin deep neural networks
  - https://arxiv.org/abs/1605.08254

- Adding gradient noise improves learning for very deep networks
  - https://arxiv.org/abs/1511.06807
- The effect of gradient noise on the energy landscape of deep networks
  - https://arxiv.org/abs/1511.06485
- The effects of adding noise during backpropagation training on a generalization performance
  - https://ieeexplore.ieee.org/document/6796981

- Accurate, large minibatch SGD: training ImageNet in 1 hour
  - https://arxiv.org/abs/1706.02677

Weight Update

- Extremely large minibatch SGD: training ResNet-50 on ImageNet in 15 minutes
  - https://arxiv.org/abs/1711.04325
- Newton's method
  - http://www.stat.cmu.edu/~ryantibs/convexopt-S15/lectures/14-newton.pdf
- An introduction to optimization chapter 9 Newton's method
  - https://www.cs.ccu.edu.tw/~wtchu/courses/2014s_OPT/Lectures/Chapter%209%20Newton%27s%20Method.pdf

- Large scale distributed deep networks
  - https://storage.googleapis.com/pub-tools-public-publication-data/pdf/40565.pdf
- Asynchronous stochastic gradient descent with delay compensation
  - https://arxiv.org/abs/1609.08326

Evaluation

Hyper Parameter Selection