Training

Arthur J. Redfern
axr180074@utdallas.edu
Oct 03, 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

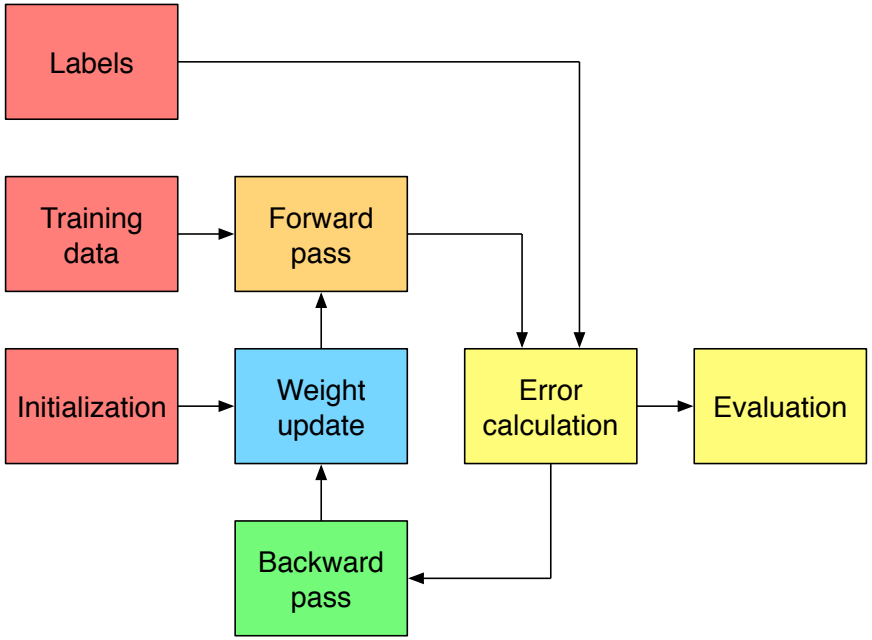
- The previous lectures looked at a lot of CNN designs
- The goal of training is to find the parameters that control the input output mapping of the (typically) linear operations in a CNN design to maximize the accuracy of the full network on testing data
- Conceptually, extracting information from training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

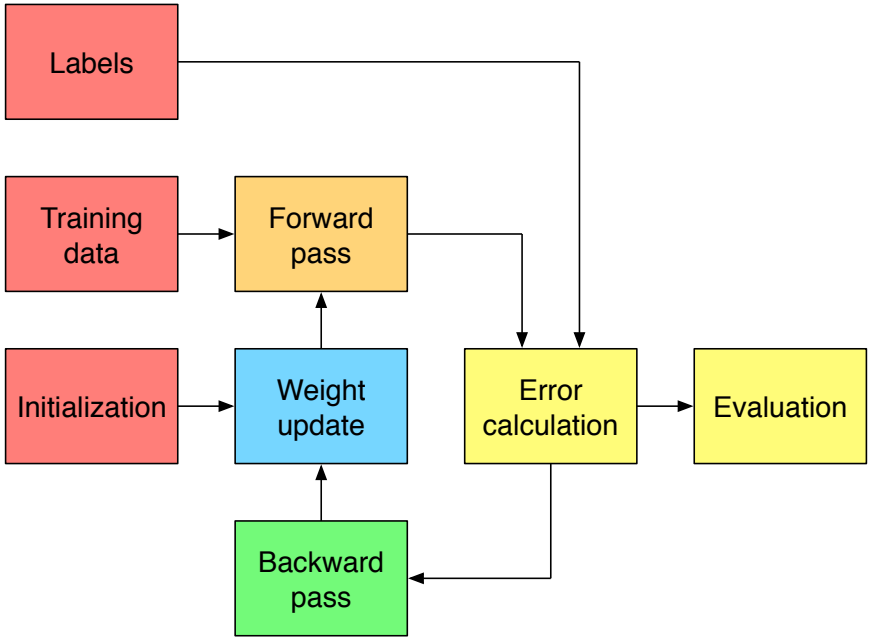


We've seen much of this already

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

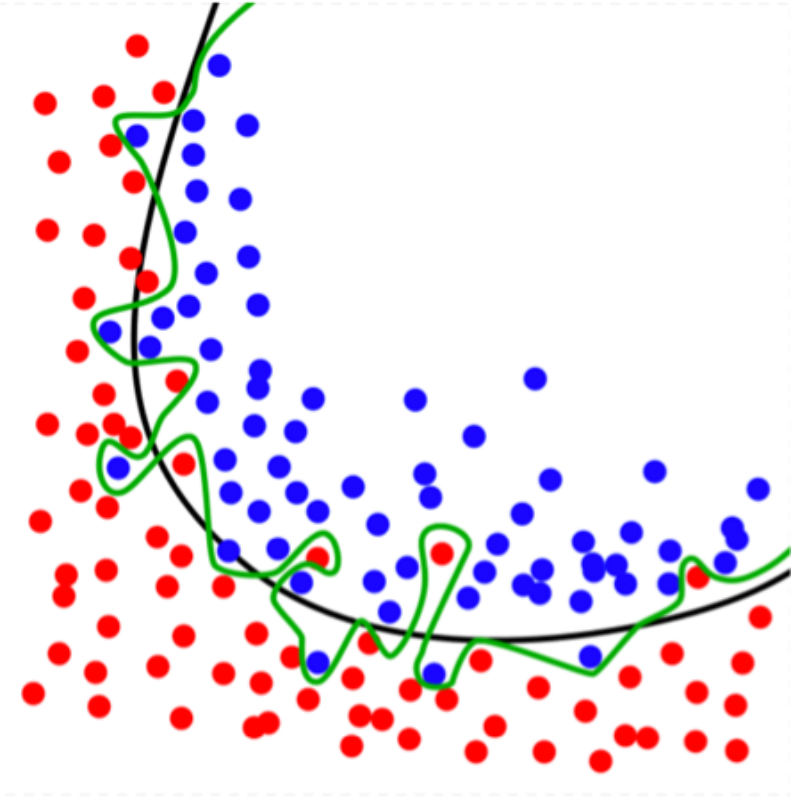


Figure from https://en.wikipedia.org/wiki/Overfitting#/media/File:Overfitting.svg

Overfitting Regularization And Generalization

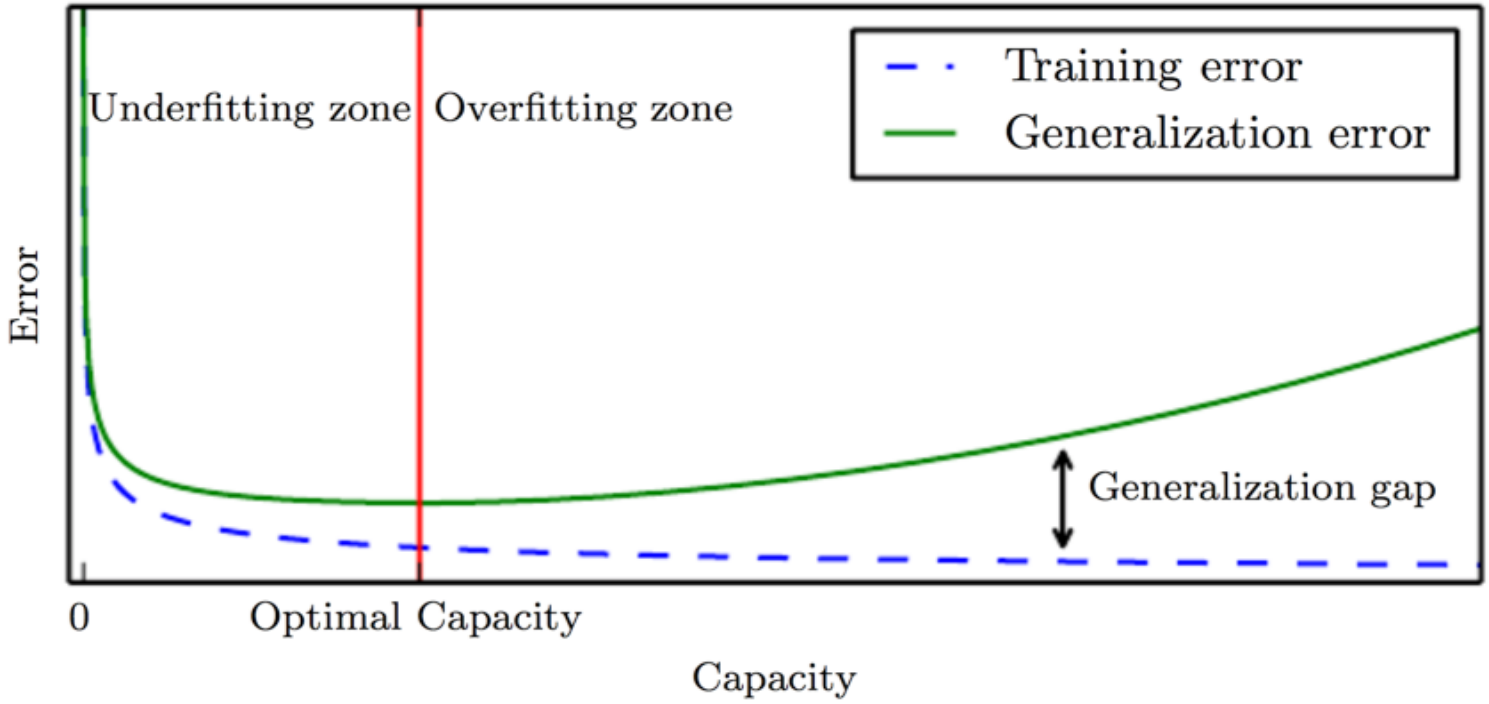


Figure from https://www.deeplearningbook.org

Convergence And Generalization

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

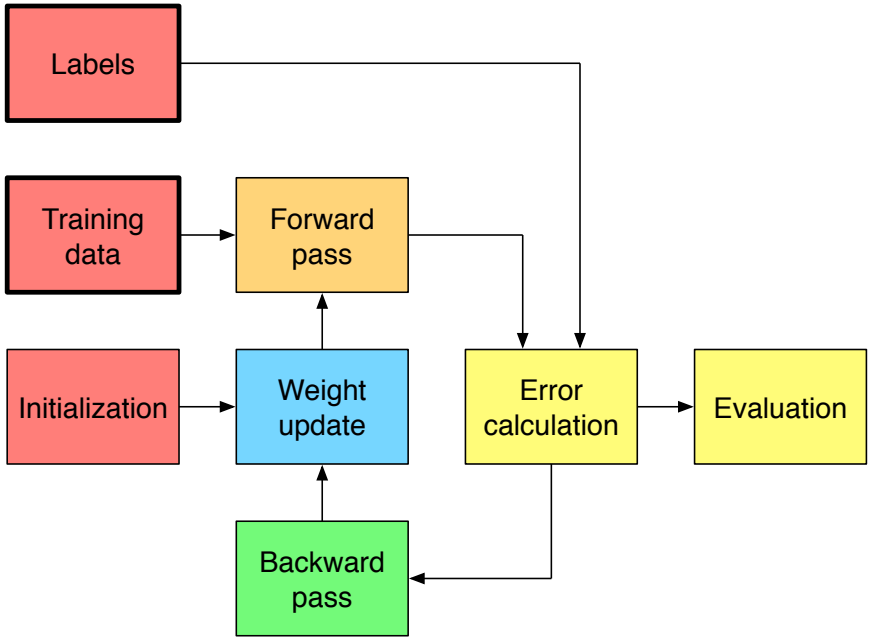
- Example: to improve convergence in very deep networks
  - Use optimal standard deviation during weight initialization
  - Add batch norms after convolutions
  - Use residual connections

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

Data

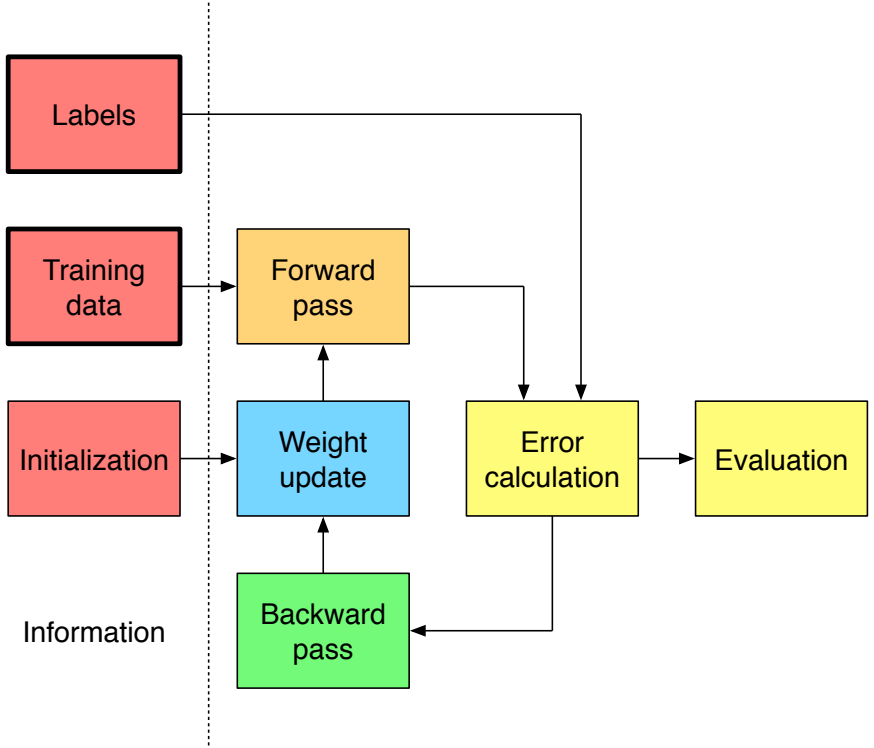
CNN Training Vs Function Optimization

- 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)

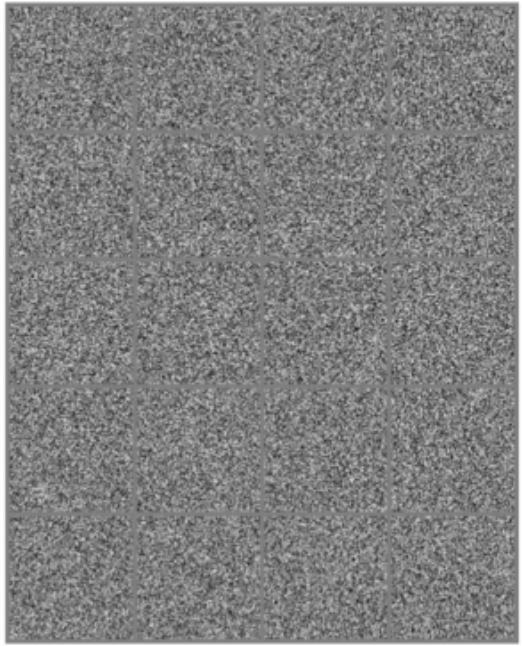


The Curse Of Dimensionality

- 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

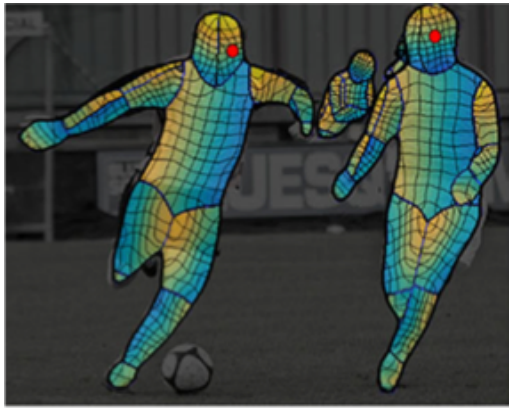
Labeling Data (Is Miserable)

- 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)

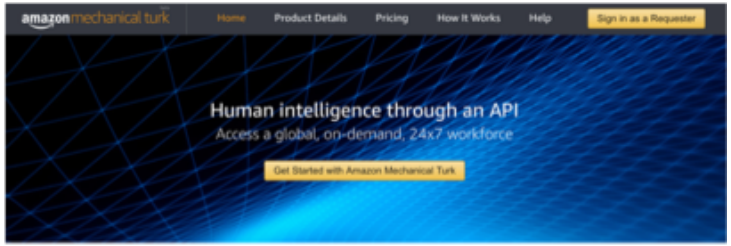
- CNNs like lots of data for training
- The complexity of labeling a single sample can vary a lot
  - Easy
    - 1 input 1 label a few classes
  - Moderate
    - 1 input 1 label many classes
  - Hard
    - 1 input many labels
    - Common to build tools to help
    - Streamline the labeling process
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



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

Tools Money Deception And Coercion

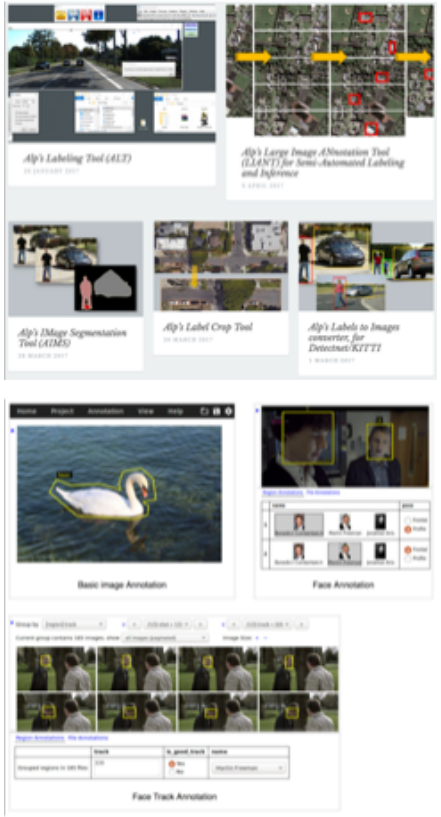
- In practice, build tools then use tools + humans to label
- How to get humans to label
  - Pay people (Mechanical Turk)
  - Coerce people (grad students, relatives)
  - Figure out a way of getting people to do it even though they don't realize they are (games, apps)
  - Realize it's not getting done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

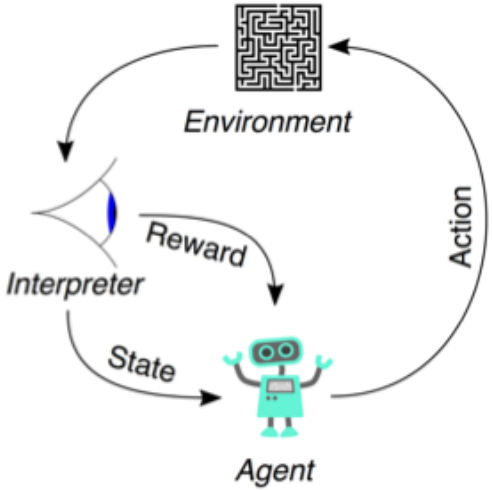
Figures from https://www.mturk.com, https://alpslabel.wordpress.com and http://www.robots.ox.ac.uk/~vgg/software/via/

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

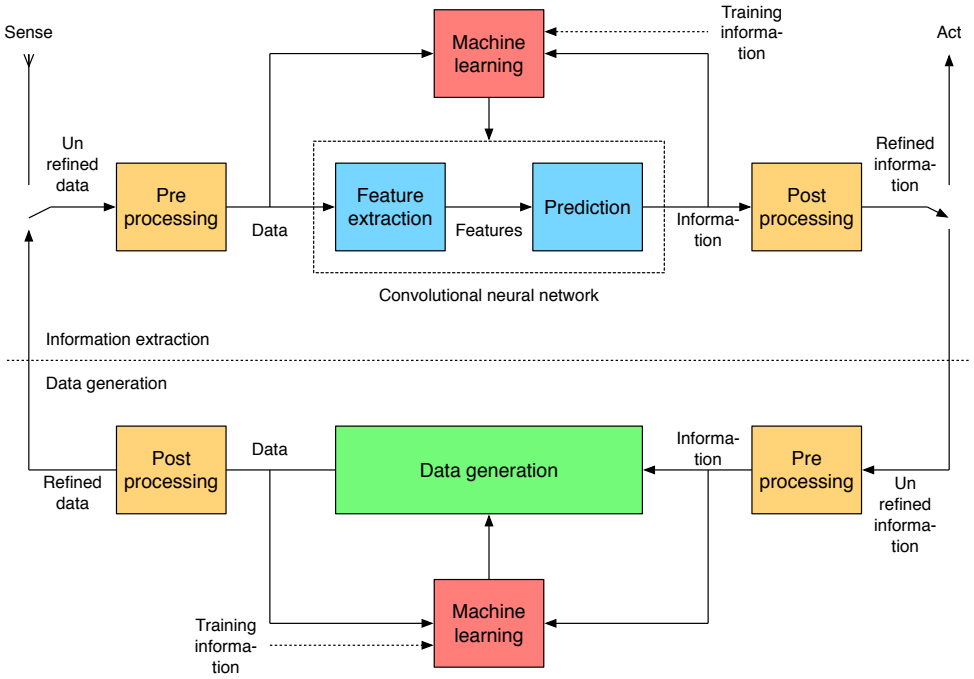
- Labeled classification datasets were created 1st for vision
  - Easiest task to outsource (e.g. Amazon Mechanical Turk)
  - Largest public dataset: ImageNet (1.2M images, 1000 classes)
  - Largest private dataset (that I know about): JFT (100M images, 15k classes)
- Pixel wise labeling can be very 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

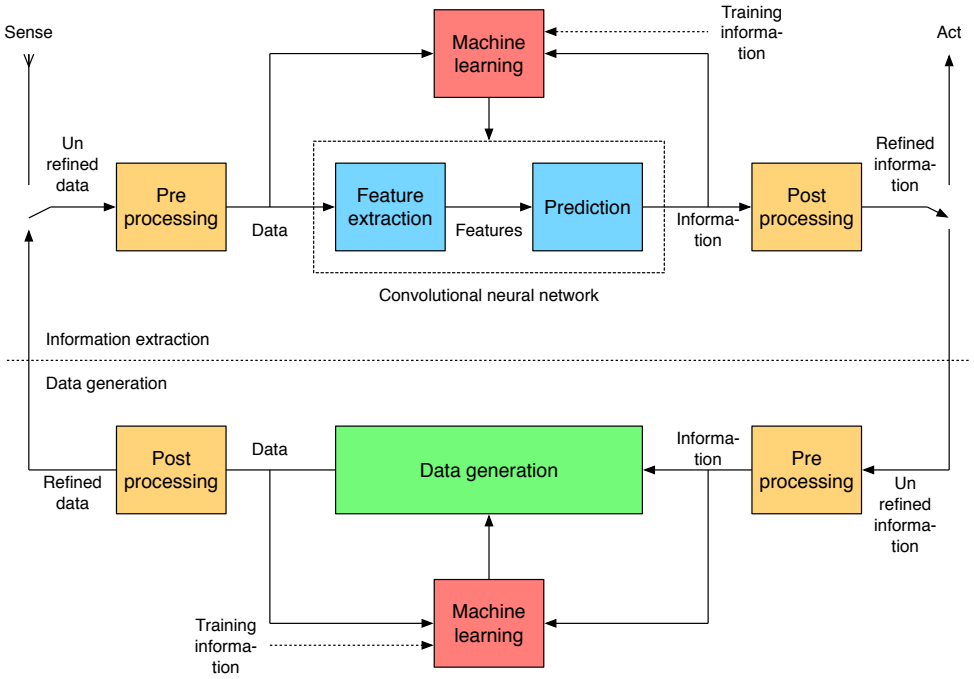
- Reminder
  - Introduction lecture mentioned that there are 2 types of problems
  - Data to information (basically everything we've talked about so far)
  - Information to data (relevant here among other places)

- 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



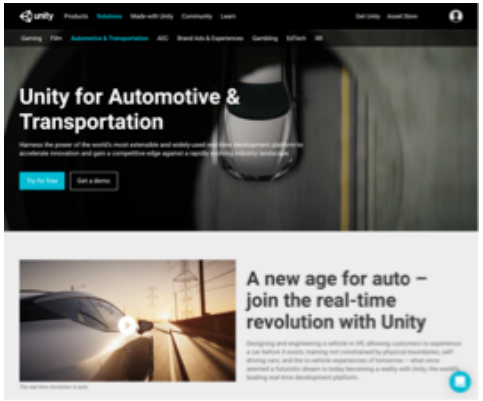
2 Ways To Generate Synthetic Data

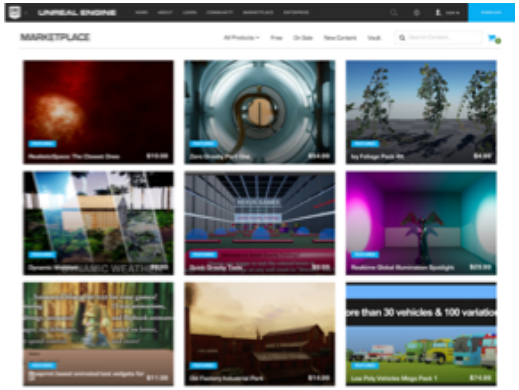
- Reminder
  - When we talked about methods for going from data to information there were 2 basic categories
    - Hand engineered
    - Learned
  - The same applies to going the other direction from information to data
- Hand engineered
  - Requires intelligence on our part
  - Ex: computer vision (thank 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)





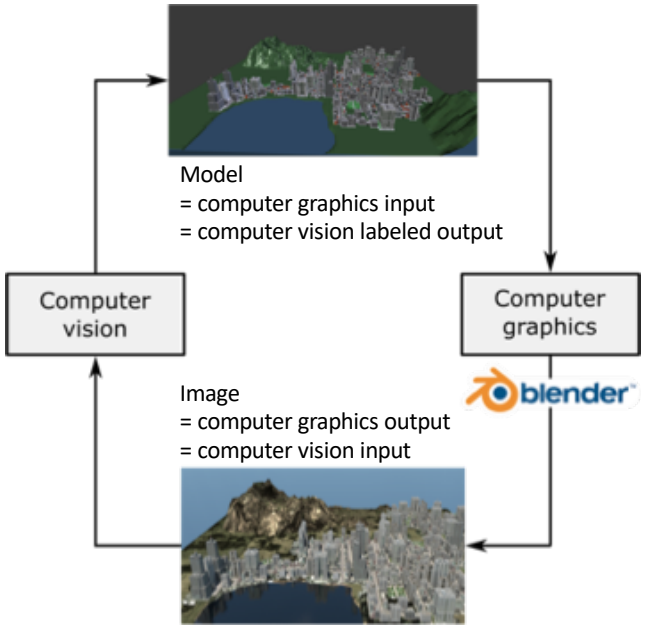
- Examples
  - Computer graphics

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

Computer Graphics

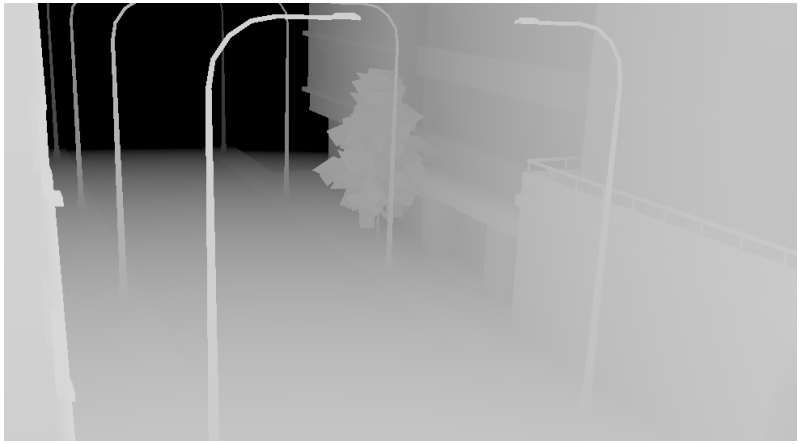
- Computer graphics and computer vision are the opposites (cousins?) of each other
  - We can take advantage of advances in computer graphics from gaming, ... to generate realistic synthetic data

- 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

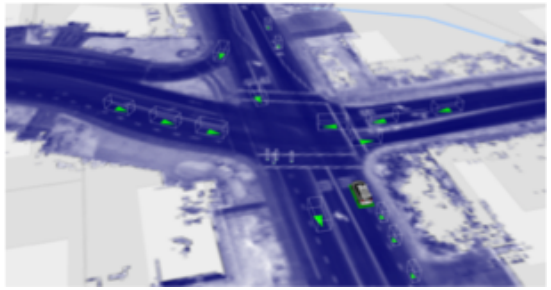
Example RGB and depth renderings
Dense depth labels are difficult to obtain in practice (e.g., KITTI is ~30% labeled)

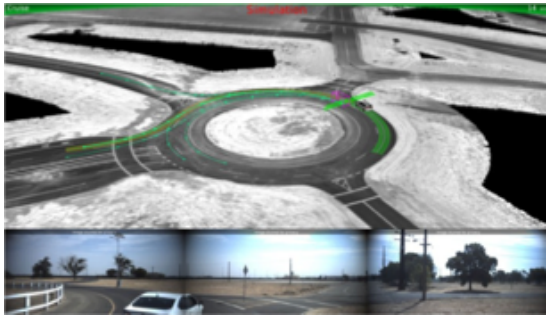
Synthetic Virtual KITTI Data

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/

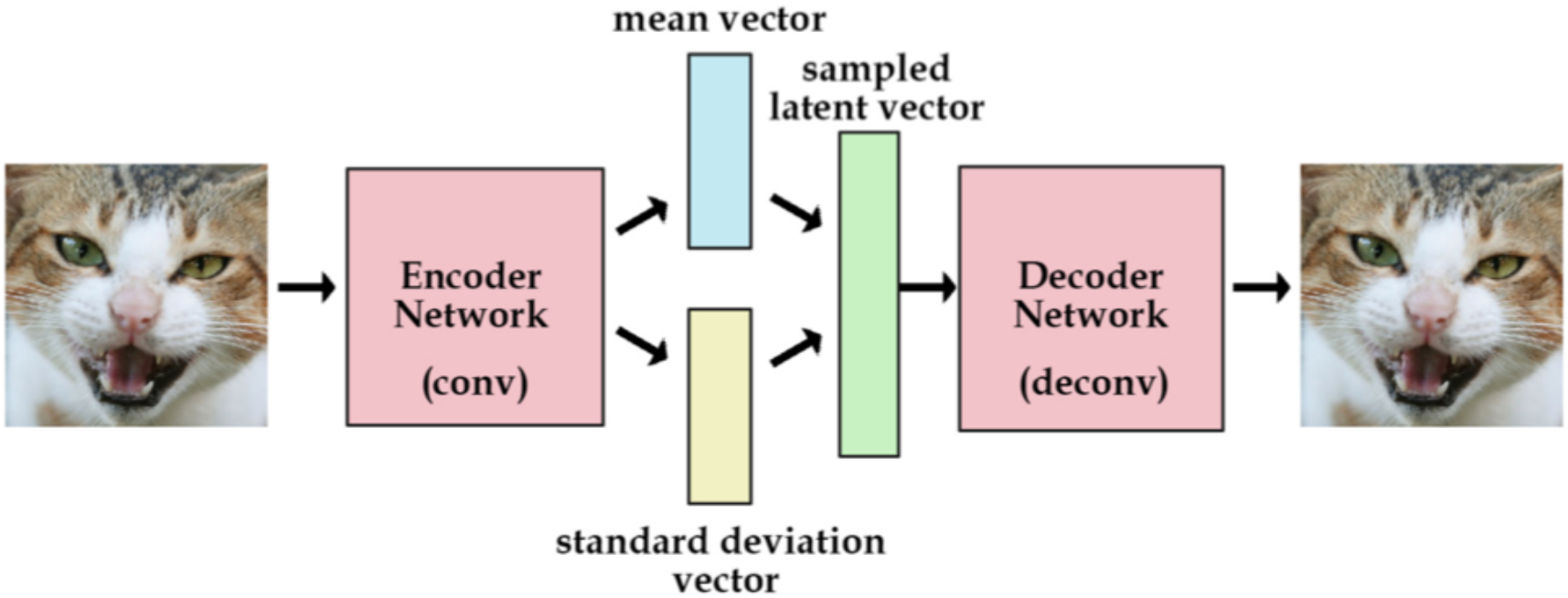
Learned

- 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)

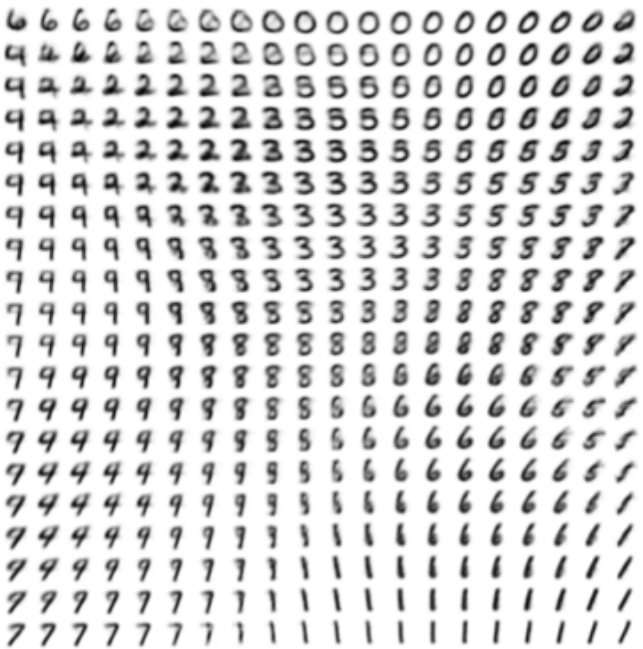
- A generative adversarial network is an example that learns a generative model that generates examples similar to the characteristics of natural samples
  - Typically learned in conjunction with another algorithm for information extraction that determines if data is natural or synthetic

Variational Auto Encoder



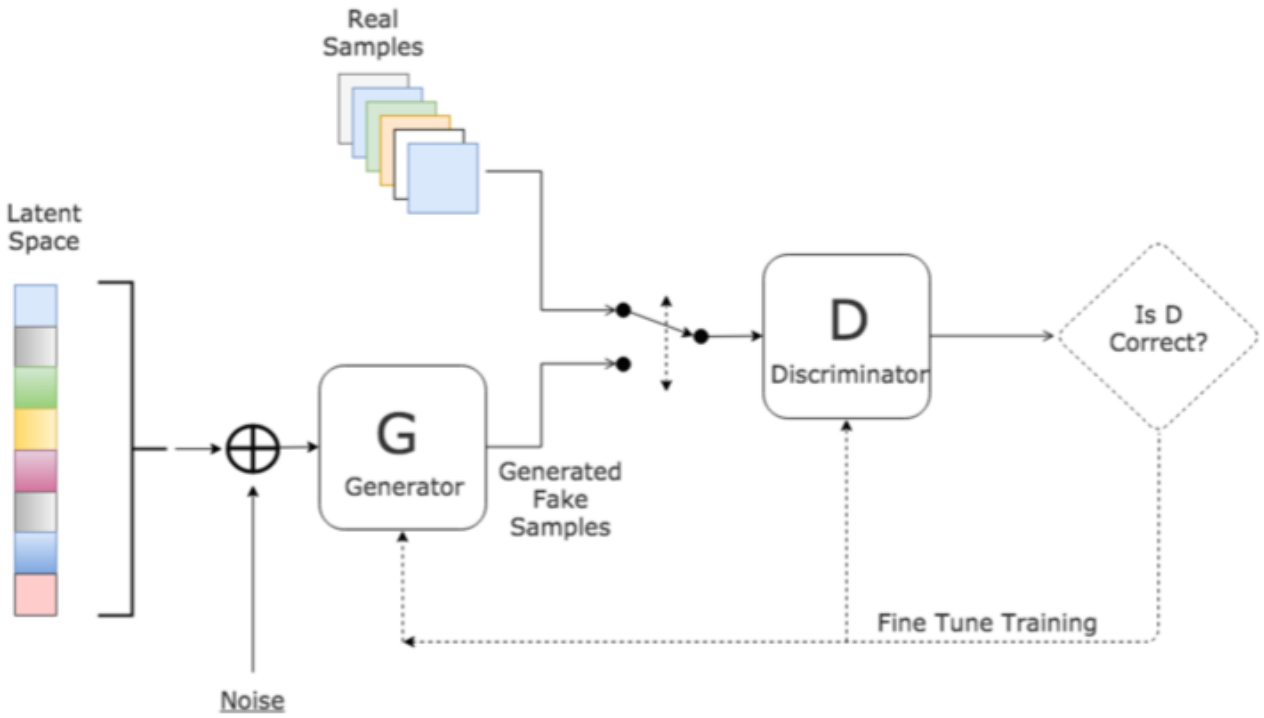
Variational autoencoders explained (http://kvfrans.com/variational-autoencoders-explained/)

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)

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