HW4: Understanding CNNs and Generative Adversarial Networks (GANs)

-This will be a “tutorial” based homework. Step-by-step instructions will be provided

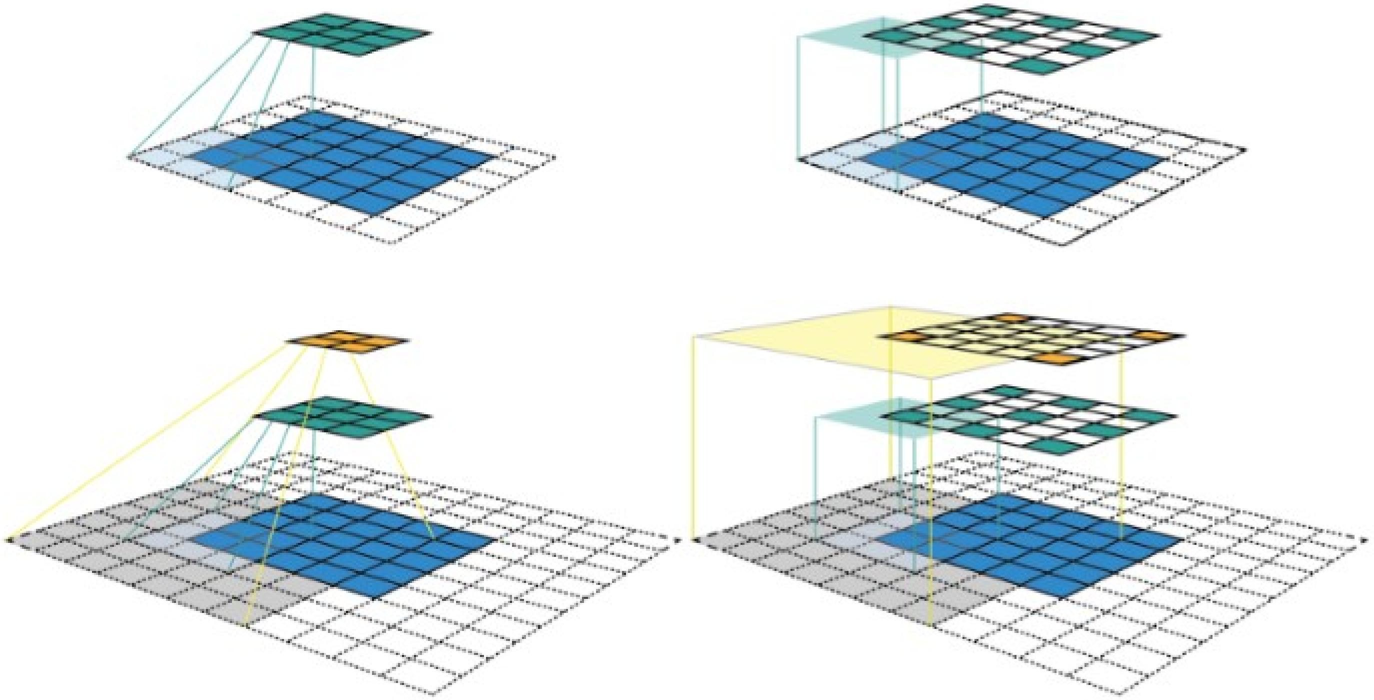
-Due Thursday, Oct 5 (next week)

-Besides training the model, most of the work can be done all at once

Understanding CNNs (What does a CNN learn?)

- Lots of questions about how many layers/hidden units/pooling/etc. - no definitive answer
- Does it learn small or large parts? Collection of small parts? Concept of “whole”? (Think about how you would describe an object to another person)
- Set up to “discriminate” between objects (tiger vs. soccerball? vs. basketball? vs. zebra?)
- Very hard to “generate” objects without tons of data to discriminate between (humans can easily do this)
- Layers of convolution go from simple to complex – prevents it from seeing too much at once (easier to learn)
- “Receptive Field” gets bigger as more layers are used

Receptive Field



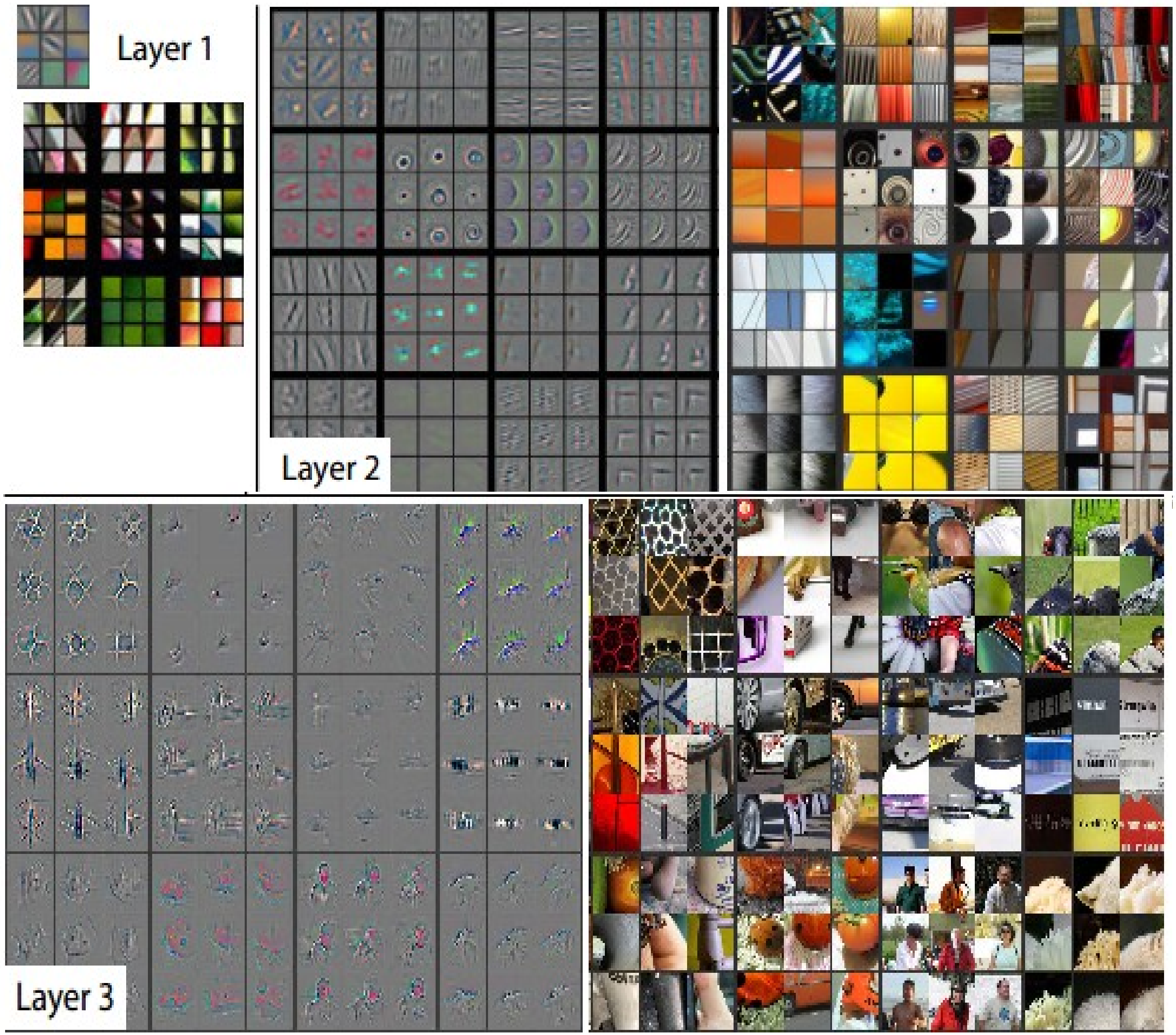
- How many pixels of the original image went into the calculation deeper within the network?
- Convolutions and pooling layers increase the receptive field
- Near the output of the network, the receptive field should be relative to the “size” of the objects of interest within the image
- Strided convolution vs. convolution + strided max pooling

https://medium.com/@Synced/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-42f33d4378e0

Network Design

- Understand your dataset to know what's necessary (big/small objects within image/distinctive features/etc.)
- Resolution of images? (CIFAR10/MNIST are small, ImageNet images are large yet most networks are cropped/resized to 224x224x3 inputs)
- Where within the picture will objects of interest be?
  - MNIST : objects aligned/scaled
  - CIFAR10 : (mostly) full object visible and takes up the whole image
  - PASCAL VOC : objects with bounding boxes (will do object detection in a later assignment)
- Can data augmentation help?

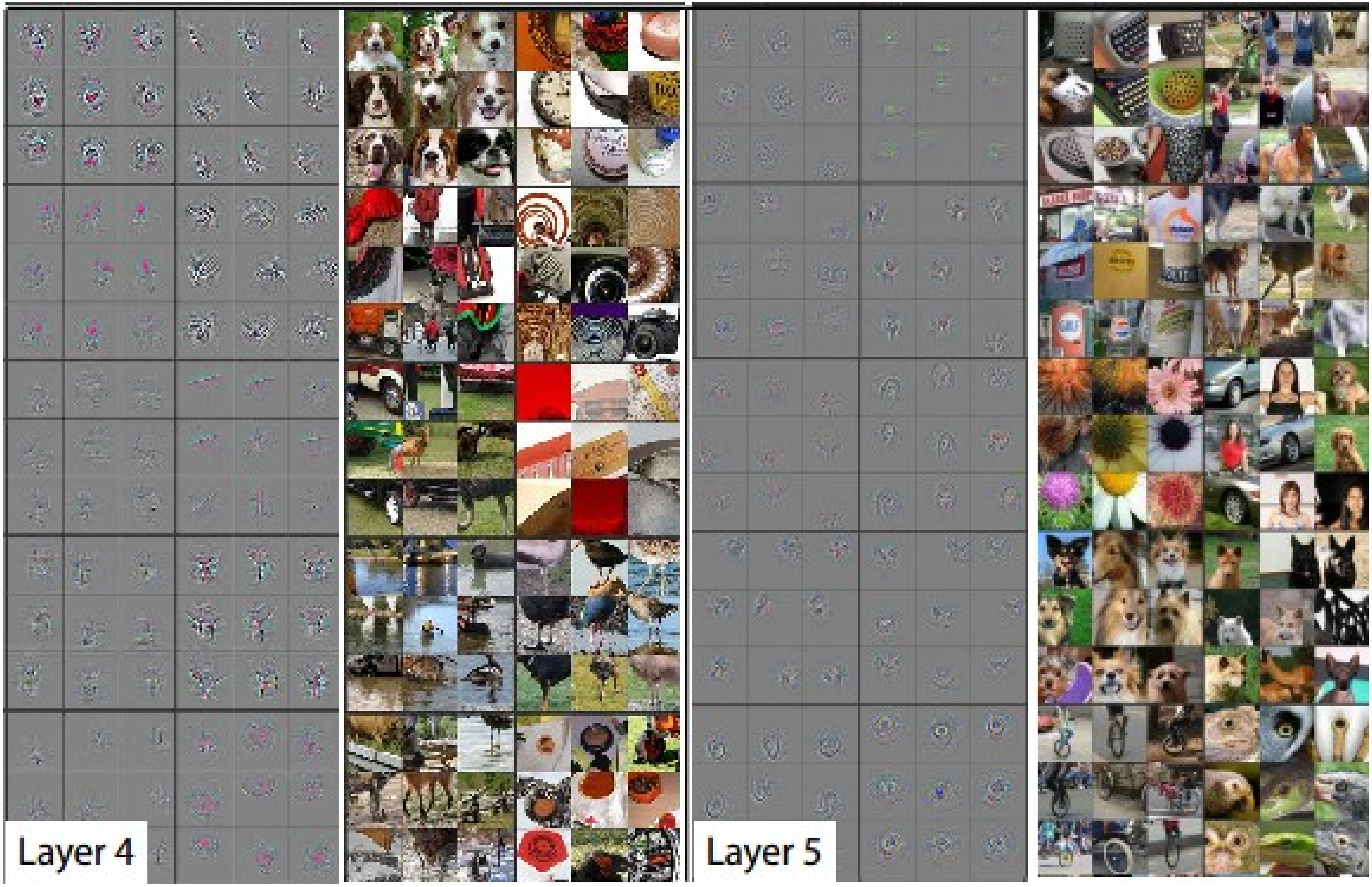
What Do Features Look Like?



- Features become more interesting and complex as network deepens
- Need lots of training data to “see” all possible types
- Overtraining – particular feature very apparent in dataset yet not necessarily representative of underlying distribution
- Detecting tanks (not sure if real story but demonstrates point well) – it technically “learned” but the data distribution did not match the true targeted distribution

Visualizing and Understanding Deep Nets - https://arxiv.org/pdf/1311.2901.pdf

What Do Features Look Like?



Visualizing and Understanding Deep Nets - https://arxiv.org/pdf/1311.2901.pdf

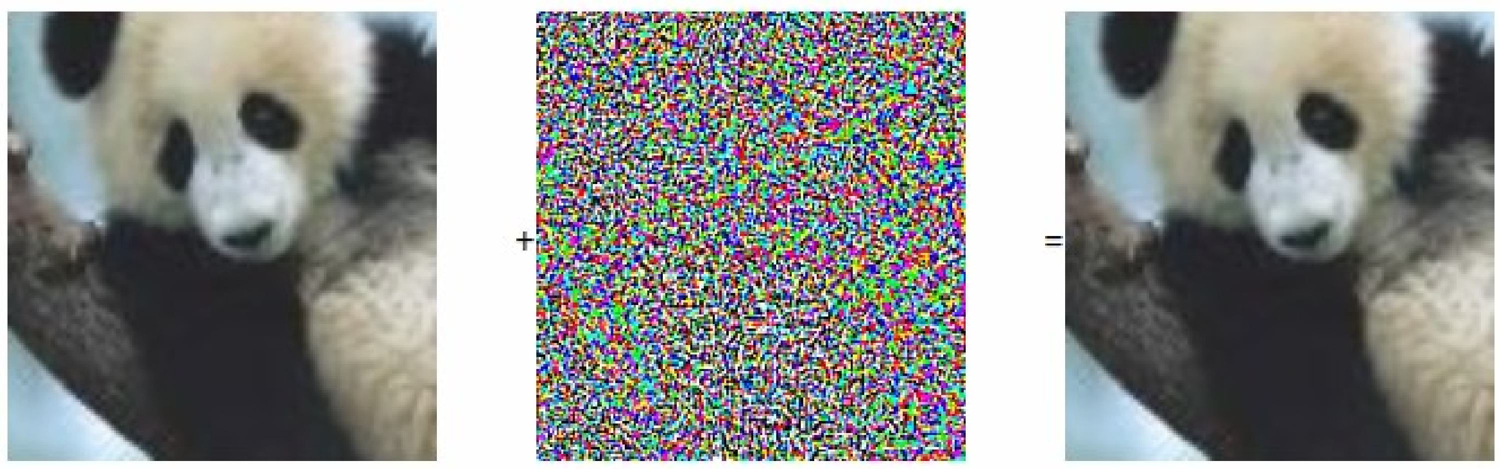
Synthetic Images with Large Class Scores

- Backpropagate error to noisy initialized image until network outputs large logit for desired class
- Images don't look real but distinctive features – multiple objects, some patterns (dalmatians), edges (computer keyboard), object size (limousine)
- Lots of random points/colors/contours that cause high activation but not what we would think is important
- You will be doing this for your assignment with CIFAR10 dataset

Deep Inside Convolutional Networks: Visualizing Image Classification Models and Saliency Maps - https://arxiv.org/pdf/1312.6034.pdf

Tricking Networks

- 8-bit images (0-255) rescaled between -1.0 and 1.0 (resolution of 0.0078)
- Backpropagate error from alternative class to real image, use sign of the gradient (-1 or 1) and multiply by 0.0078
- Models behave “too linearly”, extrapolate far from the training data and become overconfident



Original image classified as a panda with 60% confidence.

Tiny adversarial perturbation.

Imperceptibly modified image, classified as a gibbon with 99% confidence.

http://www.kdnuggets.com/2015/07/deep-learning-adversarial-examples-misconceptions.html

Tricking Networks (CIFAR10)

- Each pixel changed by +/- 10*0.0078 based on sign of the gradient
- 59/64 went from correct label to incorrect label
- You will do this on your homework

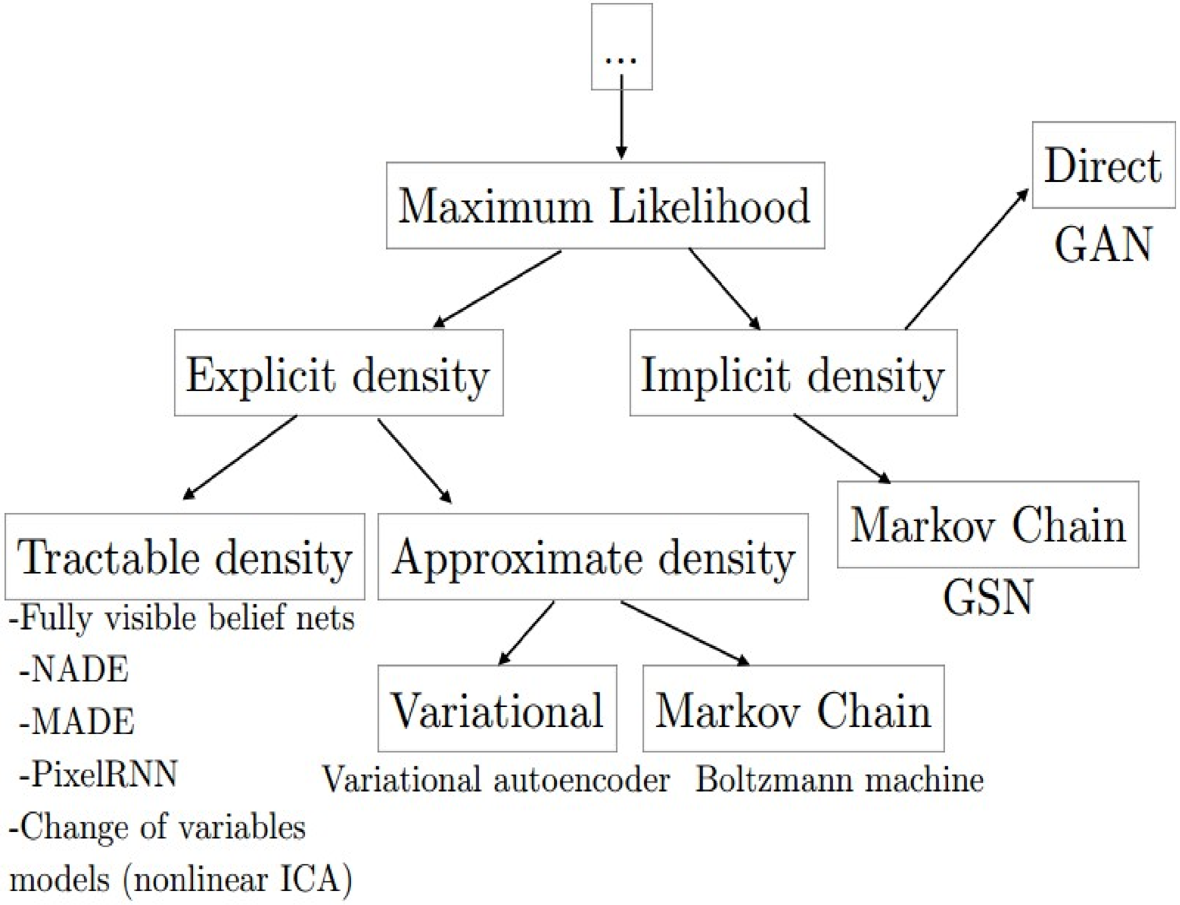
Generative Adversarial Networks (GANs)

NIPS 2016 Tutorial: Generative Adversarial Networks - https://arxiv.org/pdf/1701.00160.pdf

- Very important that everyone reads this paper, it's very easy to follow along without dense language (57 pages with lots of images)
- Near comprehensive study of GANs, mentions nearly all of the important papers
- Goes more in depth on everything I say here
- All additional pictures will be from this paper unless otherwise mentioned, check for sources

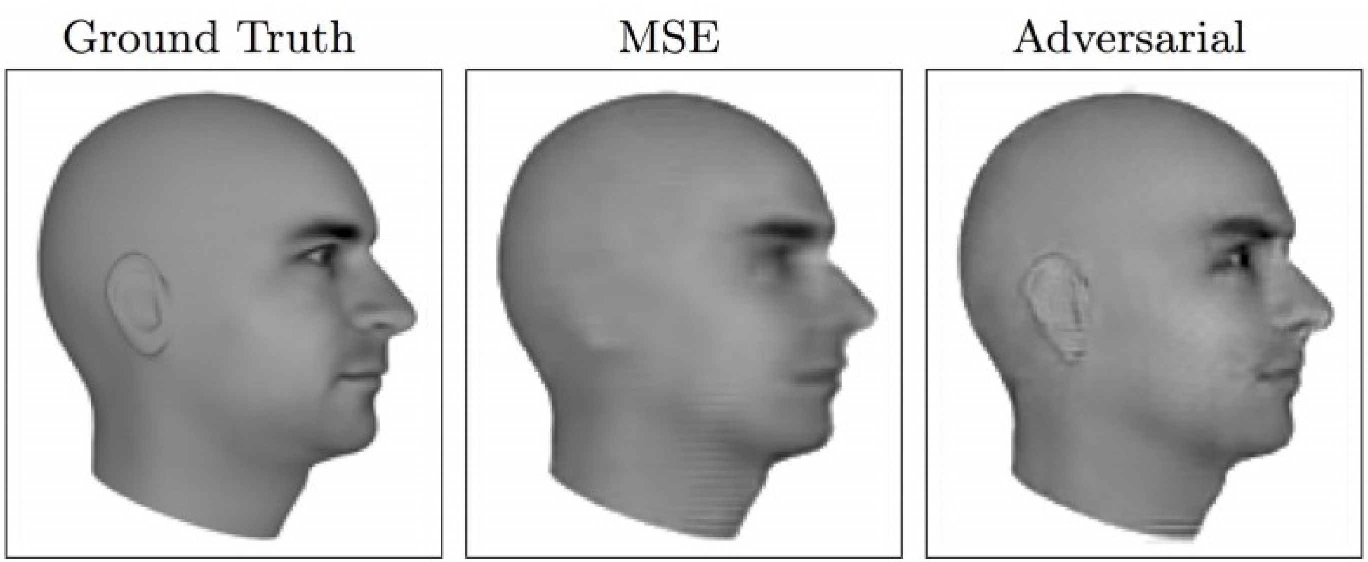
GANs

- Lots of generative models have been studied in the past
- Neural networks are discriminative models (calculates pseudo-posterior probability P(y|x))
- Most generative models map data to probability distributions (likelihood P(x|y))
- GANs are very flexible: as long as you can back propagate some error, no need to explicitly model distributions
- However, they can be very difficulty to train



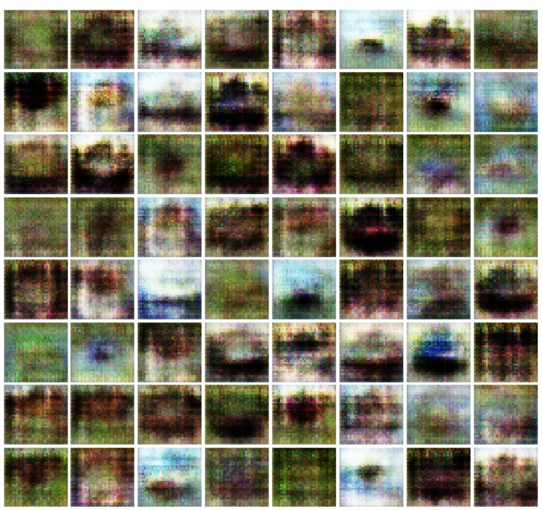
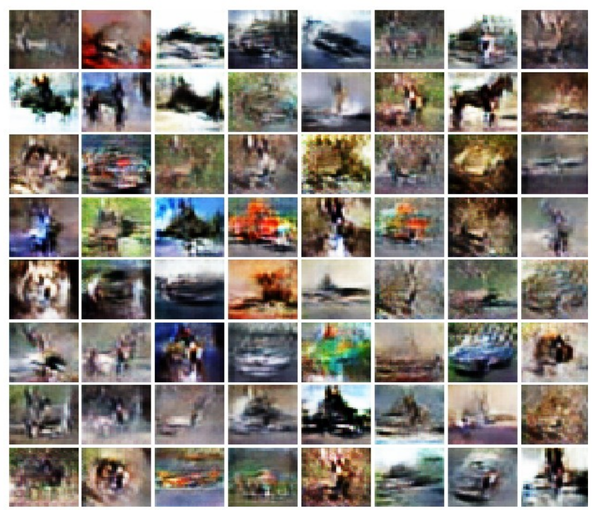
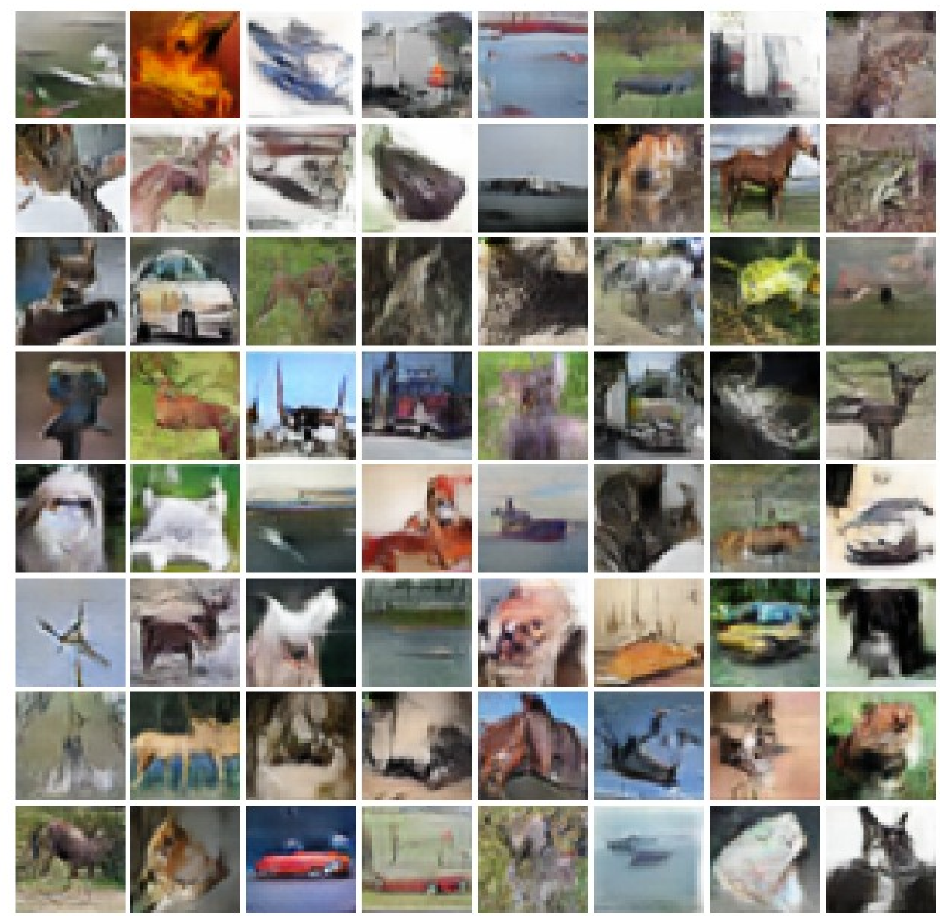
Importance of GANs

- Represent and manipulate high dimensional probability distributions
- Reinforcement learning: model based to “generate” realistic experiences to learn from
- Semi-supervised learning: easier to generate some data than hand labeling everything necessary for supervised learning, can act as a regularizer (will see in homework)
- Capable of handling multi-modal models



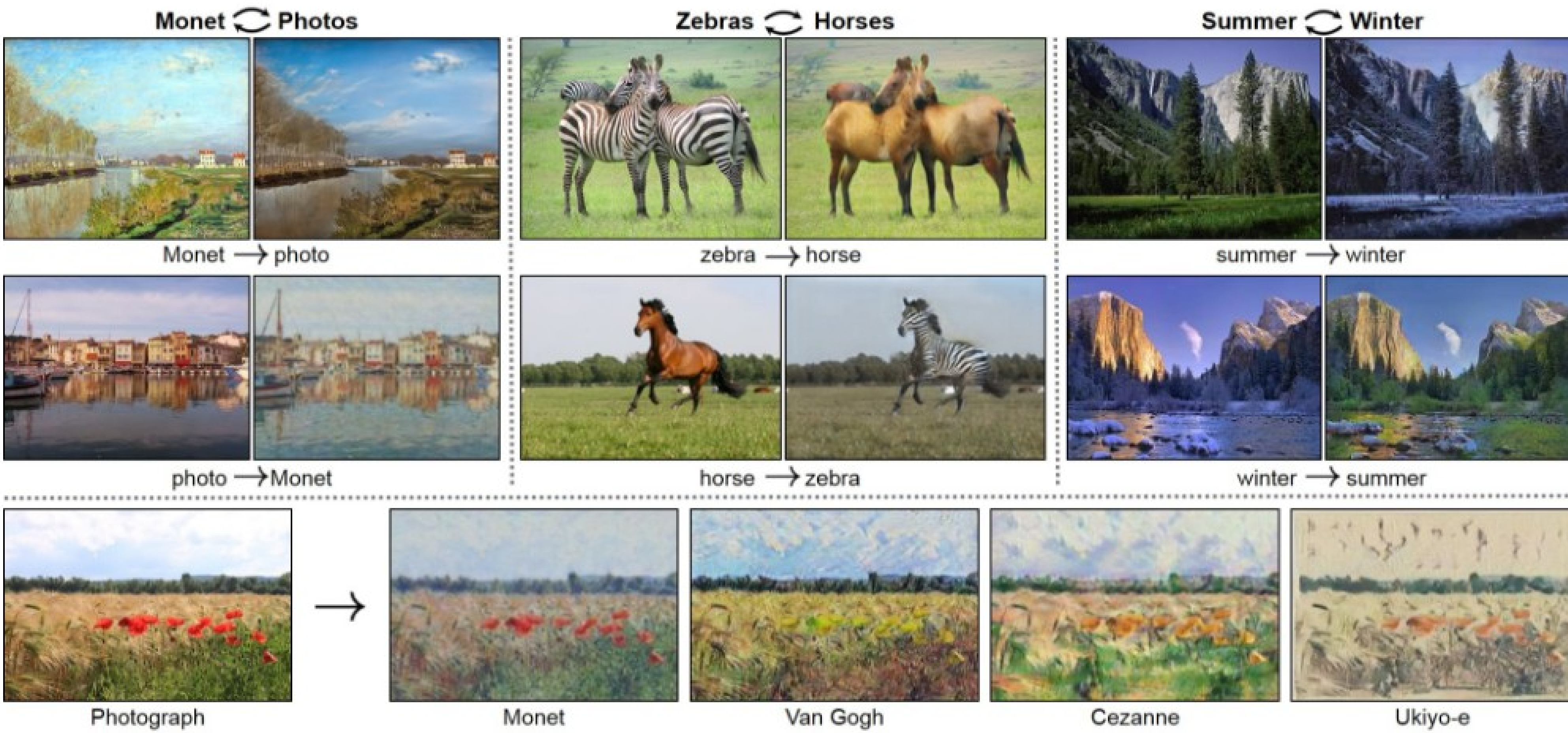
Importance of GANs

- Tasks specifically requiring generation of images
- You will train a network on CIFAR10 to do this for the HW (results below)

Importance of GANs

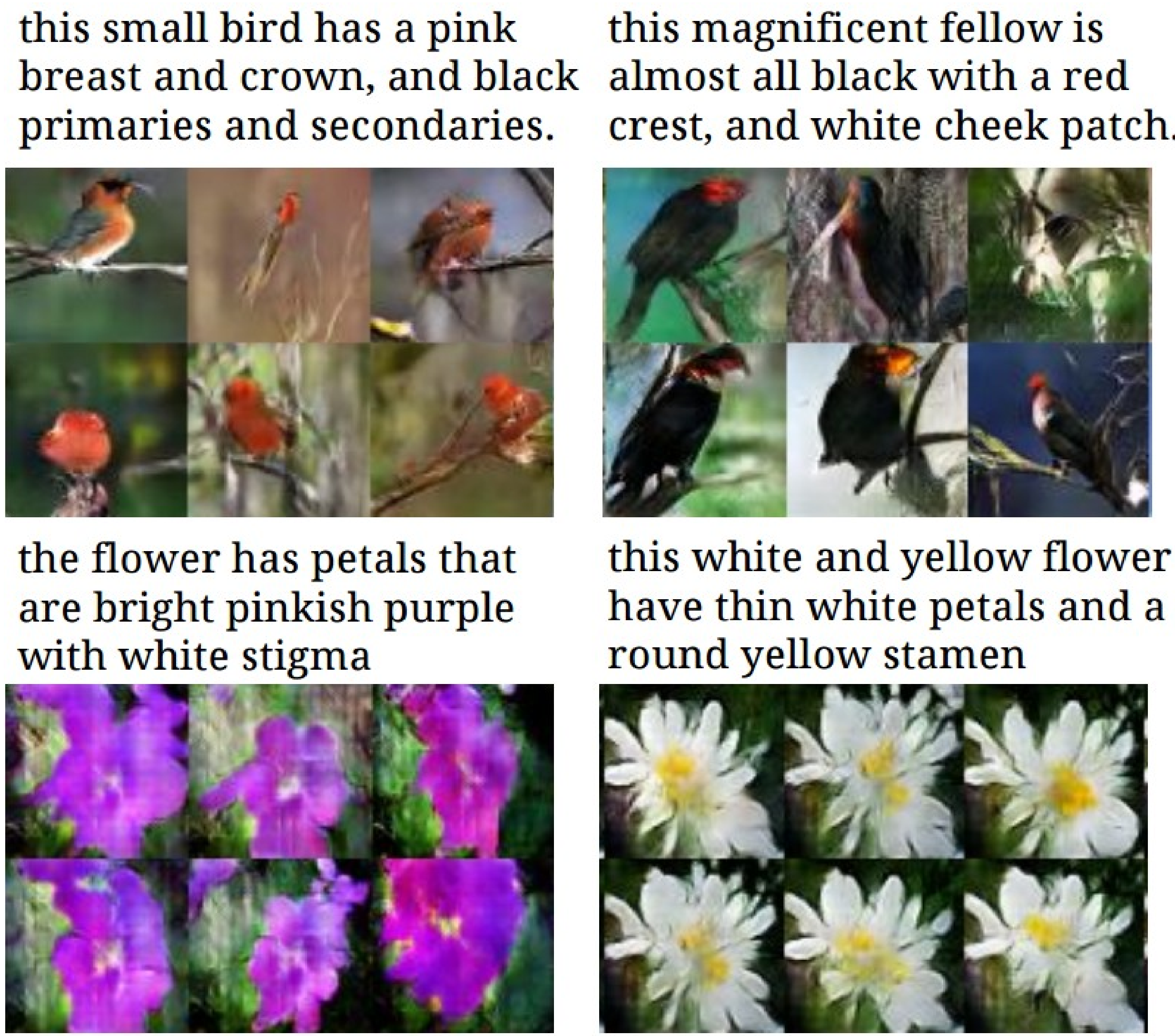
- Art (style transfer) and Image-to-Image Translation



Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks - https://junyanz.github.io/CycleGAN/

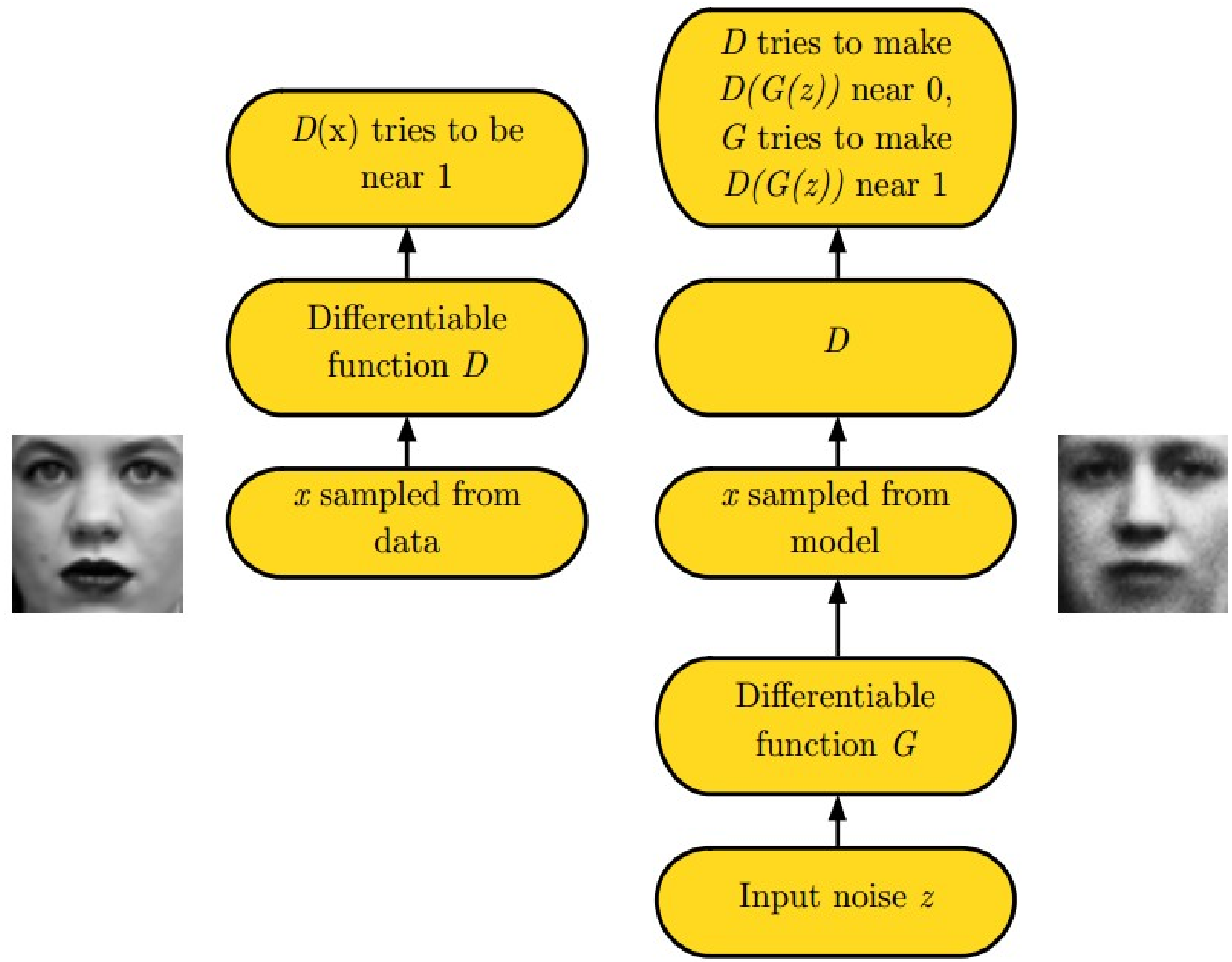
Importance of GANs

- Text-to-image synthesis



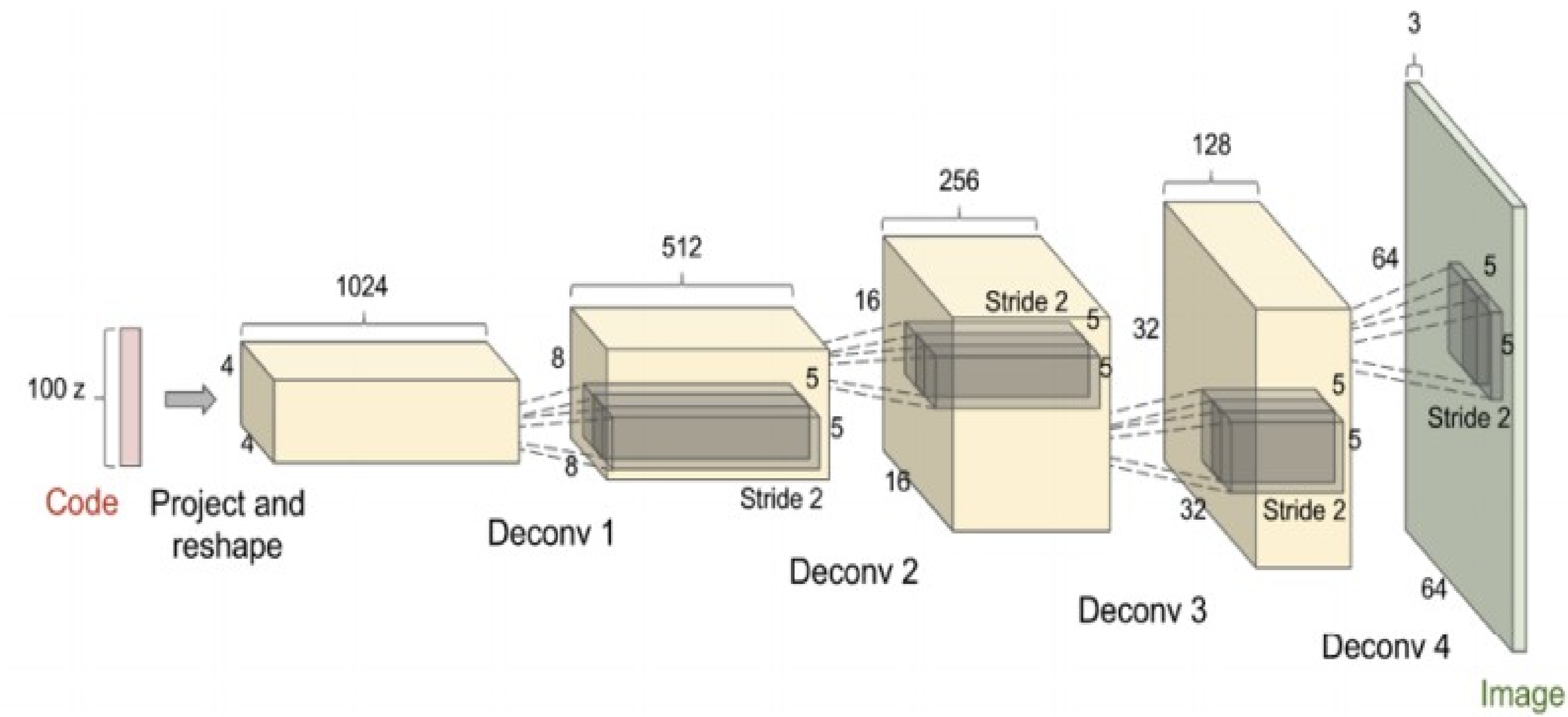
GANs Setup

- Consists of a generator network and a discriminator network
- Generator will sample a vector from a random distribution and manipulate it with transposed convolutions into a fake image
- Discriminator will be a basic convolutional network



GANs - Generator

- Transposed convolutions are typically referred to as deconvolutions (although deconvolution is not the correct term)
- Manipulates a low dimensional random distribution into the shape of an image



Transposed Convolution Demo - https://distill.pub/2016/deconv-checkerboard/

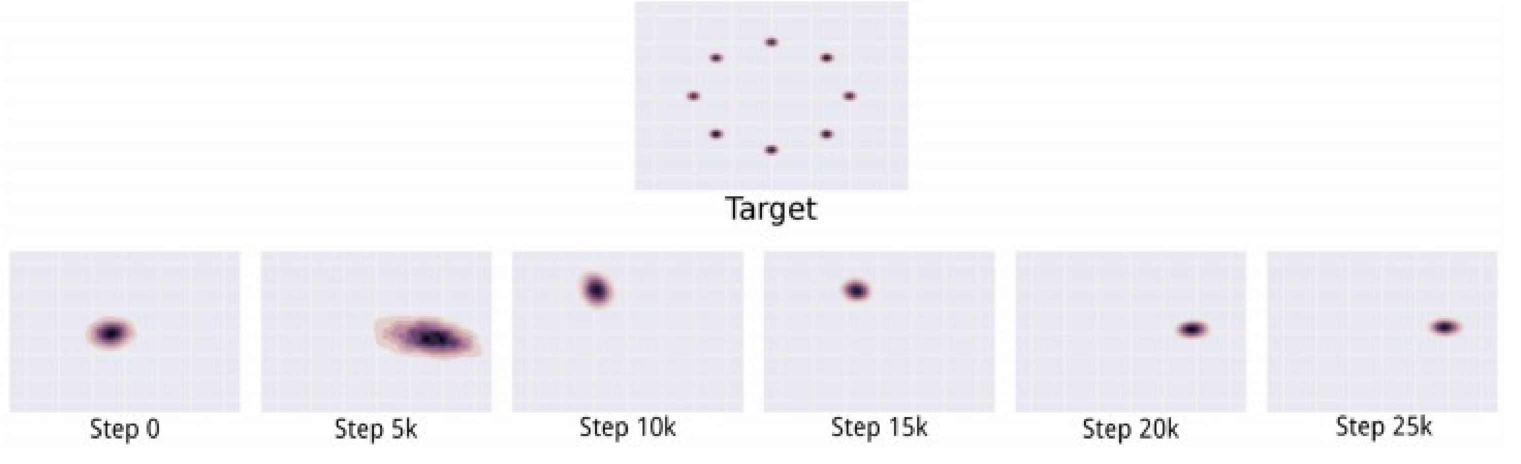
Training GAN in Tensorflow

- Generator
- Discriminator
- Loss Functions
- Training steps (separate for G and D)
- Batch Normalization (can be tricky)
- Dropout in D (prevent D from learning too quickly)
- Nearly all operations are convolution (as opposed to fully connected layers)
- Leaky ReLU
- No max pooling (use strided convolutions instead)
- Train with labels instead of just 0(real)/1(fake)
- One sided label smoothing (prevent extreme extrapolation)
- Gradient of G can vanish if D becomes too good too fast at recognizing fake images
- Instructions for implementing all of this will be detailed in the HW pdf, it is written as more of a tutorial as opposed to figuring out how to do it all on your own
- Code demonstration

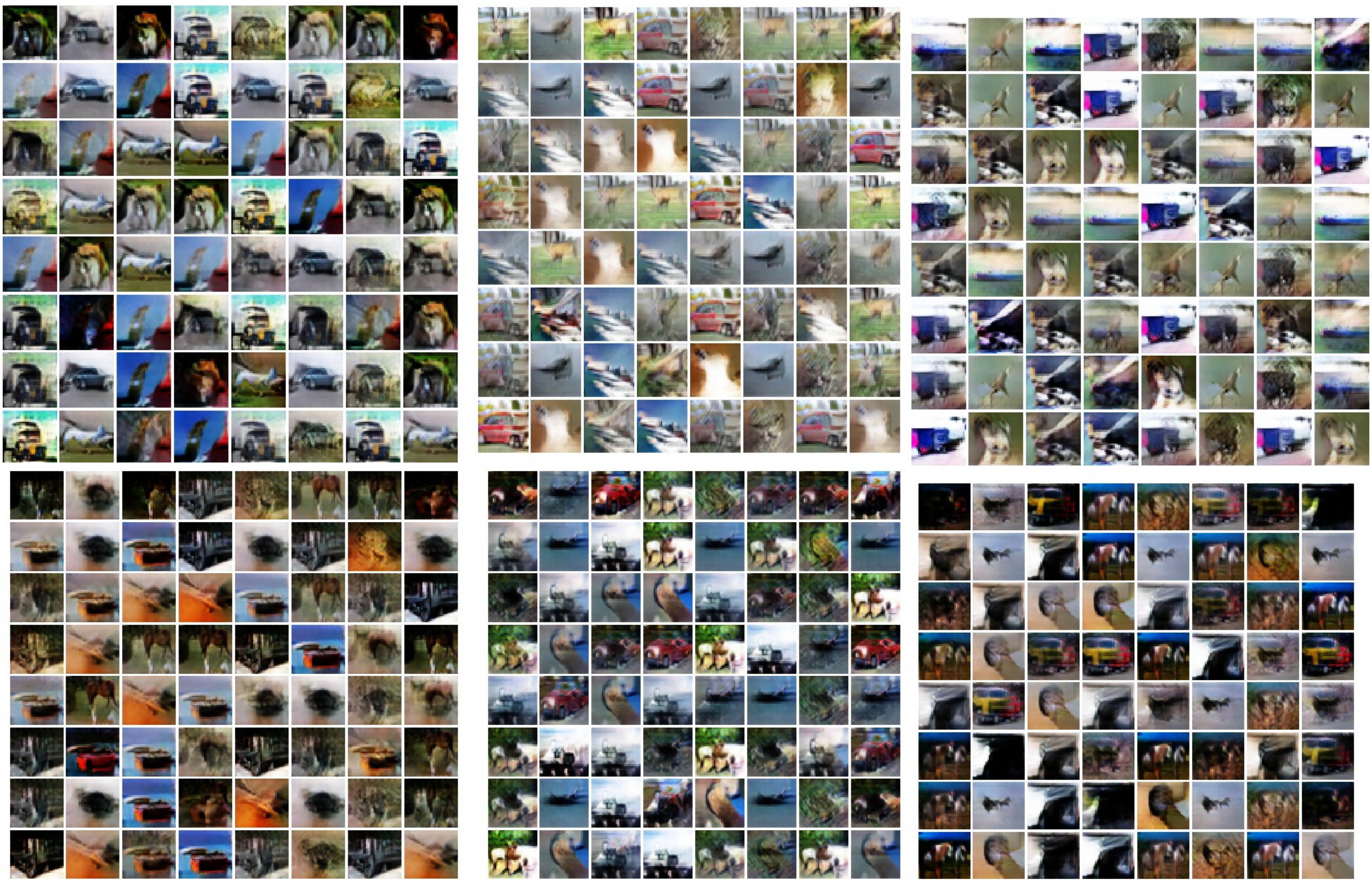
https://github.com/soumith/ganhacks - tips and tricks

Problems with GAN

- Non-convergence: Requires finding an equilibrium between two players (G and D) in a game. Sometimes can repeatedly undo progress (D focuses too much on G without learning worthwhile features, D becomes too good at recognizing fake images and G gets stuck)
- Mode Collapse – several different input values **z** mapped to the same output, the generator then oscillates between modes without further progress



Mode Collapse



Examples - Minibatch GANs

- Method includes features based on the distance between samples in a mini-batch to encourage them to be different







Examples – Trouble with Counting

- Discriminator can output a very high score because of the small part (face) without considering the object as a whole






Examples – More image-to-image translation

Examples – Face Generation

https://github.com/carpedm20/DCGAN-tensorflow

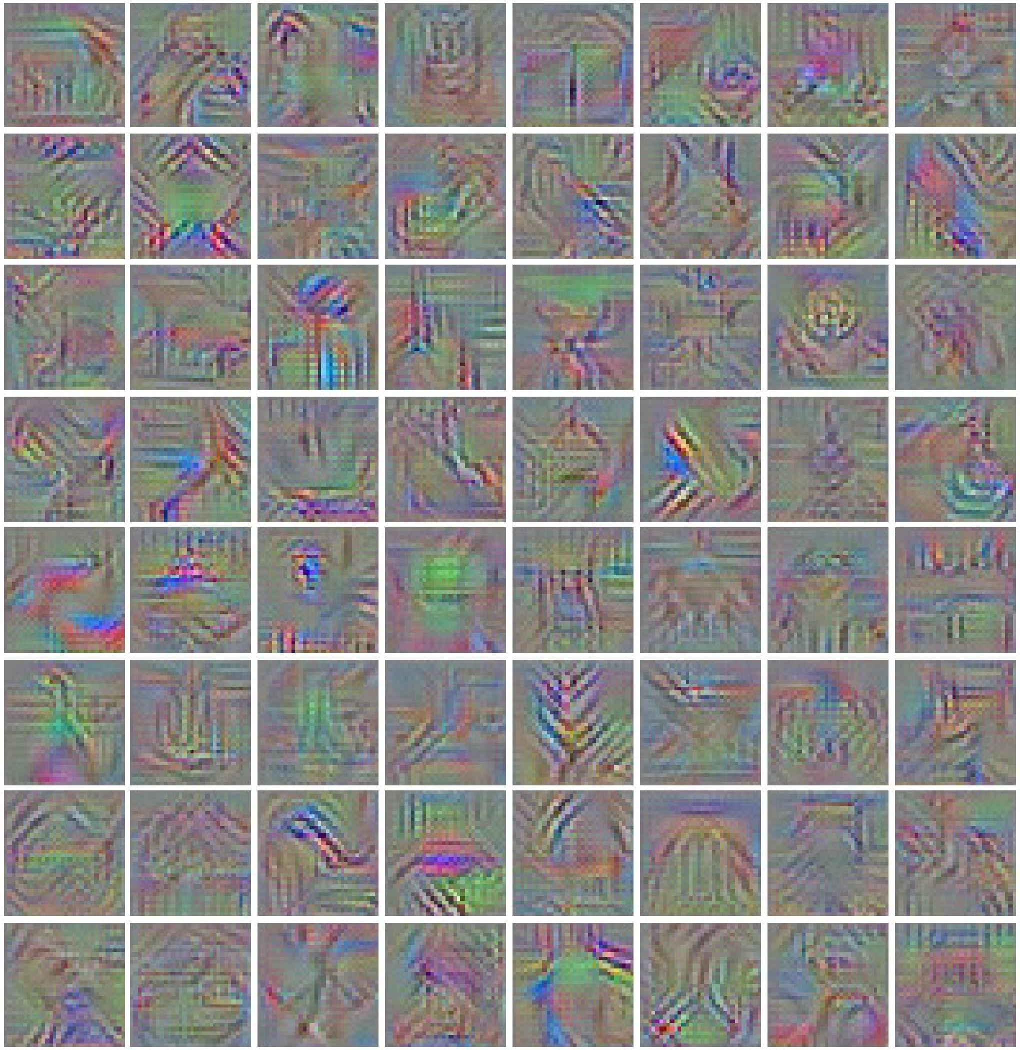
Homework

- I will provide you with a pretrained network on CIFAR10, with it you will:
  - Trick the network with slightly altered real images
  - Trick the network with slightly altered noisy images
  - Back-propagate error to create synthetic images maximizing the activation of last layer features (before output)
  - Back-propagate error to create synthetic images maximizing the action of the class output layer
- Train a GAN (generator and discriminator) on CIFAR10
  - Create animated GIF demonstrating the evolution of the generator output
  - (Hopefully) get higher accuracy on the test dataset with the discriminator trained with the GAN as opposed to the network I give you (80-81% compared to 83-85%)
- Repeat the first part using your discriminator trained with the GAN and compare with the pretrained model I provided

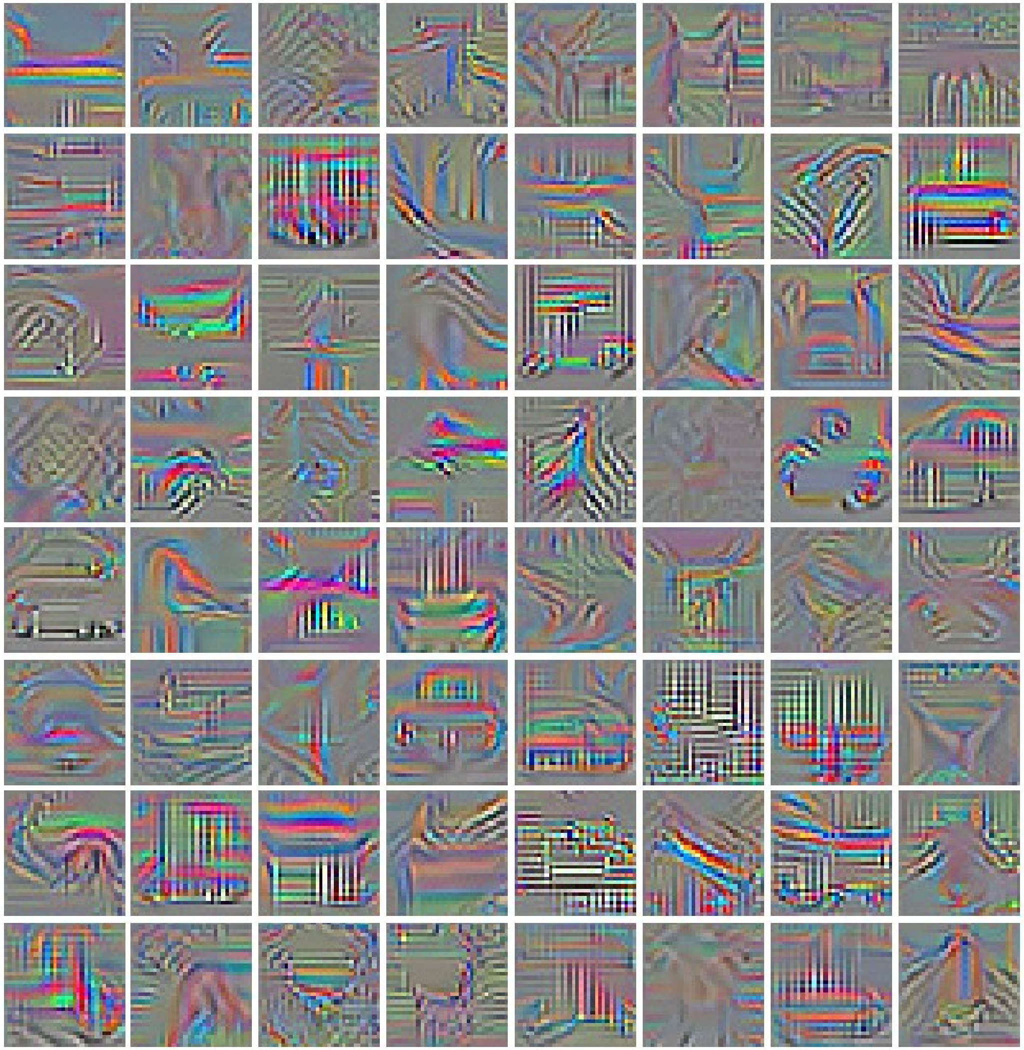
- Possible extra credit options:
  - find set of hyperparameters and a large network structure (9+ convolutional layers for D) achieving 88-90%+ accuracy and doesn't experience mode collapse
  - Write your own TF code implementing someone else's work (plenty to be found online and seen throughout this presentation), need to write a summary of your code and what it's based on so it's obvious it's not completely copied
  - https://github.com/carpedm20/DiscoGAN-pytorch Great example of pytorch code for lots of datasets that you could rewrite as tensorflow code
  - Modify code from this assignment to include feature matching and minibatch discrimination (from this paper) https://arxiv.org/pdf/1606.03498.pdf

Example HW Output

- No GAN

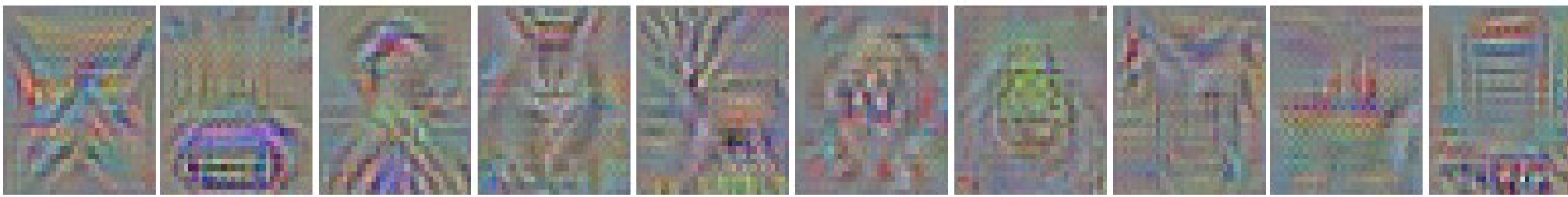


- GAN

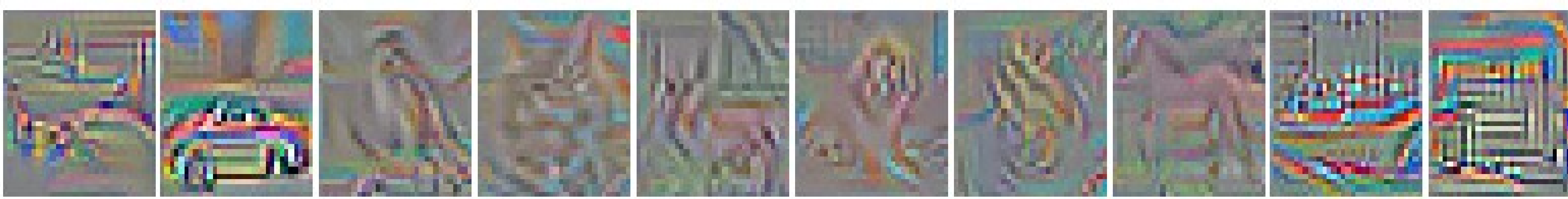


Example HW Output

- No GAN



- GAN



- Airplane car bird cat deer dog frog horse ship building

Takeaway

- Help build your intuition of CNNs to better design networks based on your data as opposed to treating it as a black box
- Help realize various problems and creative solutions neural networks can tackle as opposed to being limited by a simple classification structure
- Lots of project ideas mentioned throughout (fair warning: these can be a complete pain to train and tune, metric for success can sometimes be very subjective requiring careful monitoring of the network to understand why it's succeeding or more often failing)
- Please ask questions and let me know if anything is unclear

Some Links I had Saved but Didn't Mention

https://hardikbansal.github.io/CycleGANBlog/

http://www.inference.vc/instance-noise-a-trick-for-stabilising-gan-training/

http://yosinski.com/media/papers/Yosinski__2015__ICML_DL__Understanding_Neural_Networks_Through_Deep_Visualization__.pdf

https://distill.pub/2016/deconv-checkerboard/

https://arxiv.org/pdf/1412.6572.pdf

https://github.com/carpedm20/DCGAN-tensorflow

https://github.com/carpedm20/BEGAN-tensorflow

https://github.com/carpedm20/simulated-unsupervised-tensorflow

https://github.com/zsdonghao/dcgan