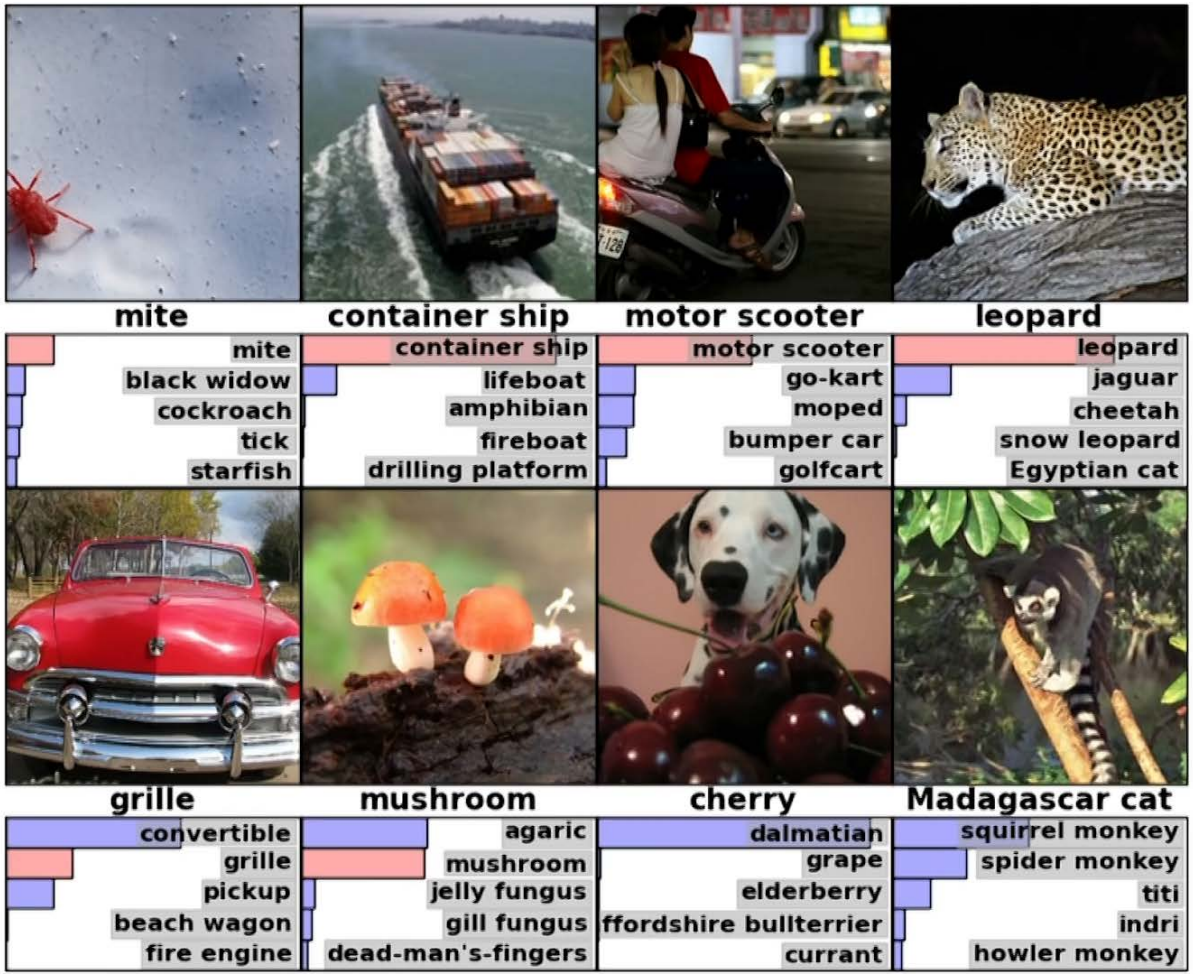
COMP 4752
Computational Intelligence

Lecture 20

Deep Neural Networks

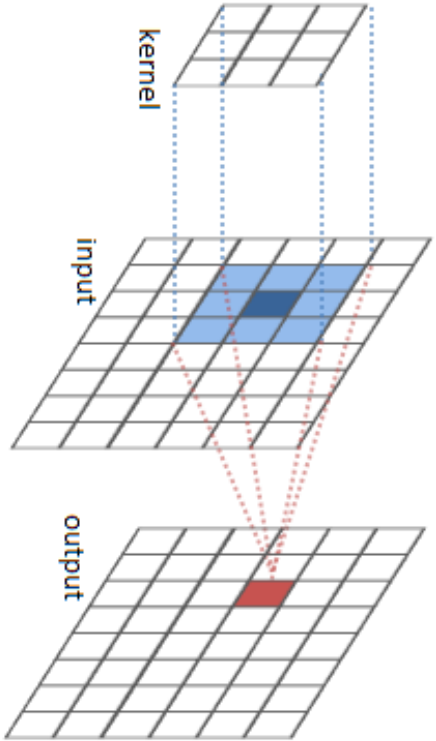
Deep Neural Networks

- Neural networks that have many hidden layers, and extra processing
- Have been popular since 2012 (Hinton)
- Rise of GPU computing power has made them feasible (floating point calcs)


mite
container ship
motor scooter
leopard
mite
black widow
cockroach
tick
starfish
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
go-kart
moped
bumper car
golfcart
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
mushroom
cherry
Madagascar cat
convertible
grille
pickup
beach wagon
fire engine
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
dalmatian
grape
elderberry
ffordshire bullterrier
currant
squirrel monkey
spider monkey
titi
indri
howler monkey

Convolutions

- Have some input image
- Have a neuron that looks at say, a 10x10 pixel area and produces an output
- That area called a Kernel
- Next neuron looks at a small shift of the area



Max Pooling

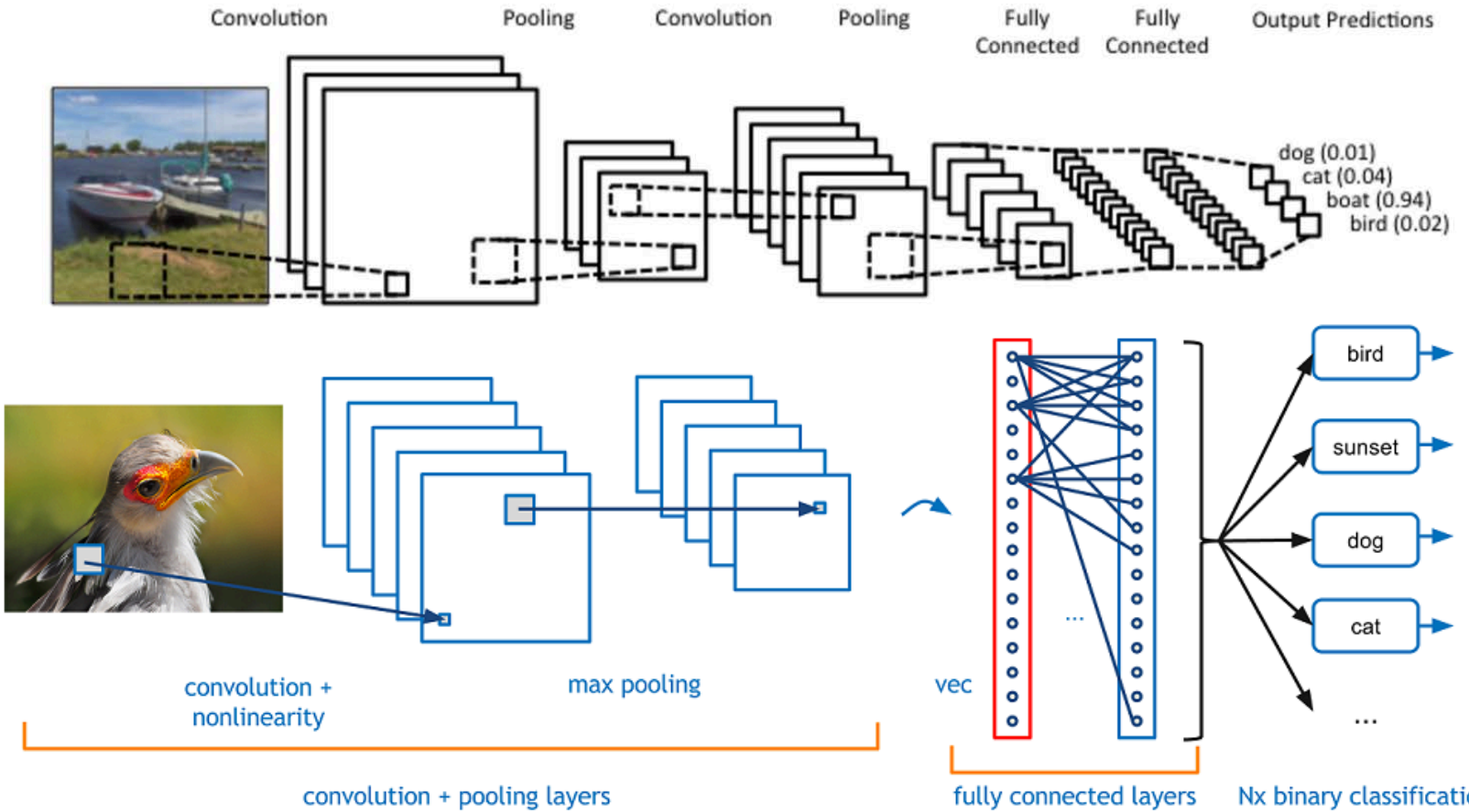
- Output from convolution produces another matrix
- Now take maximums from areas of that matrix
- This forms another matrix, the process is called pooling

12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

20	30
112	37

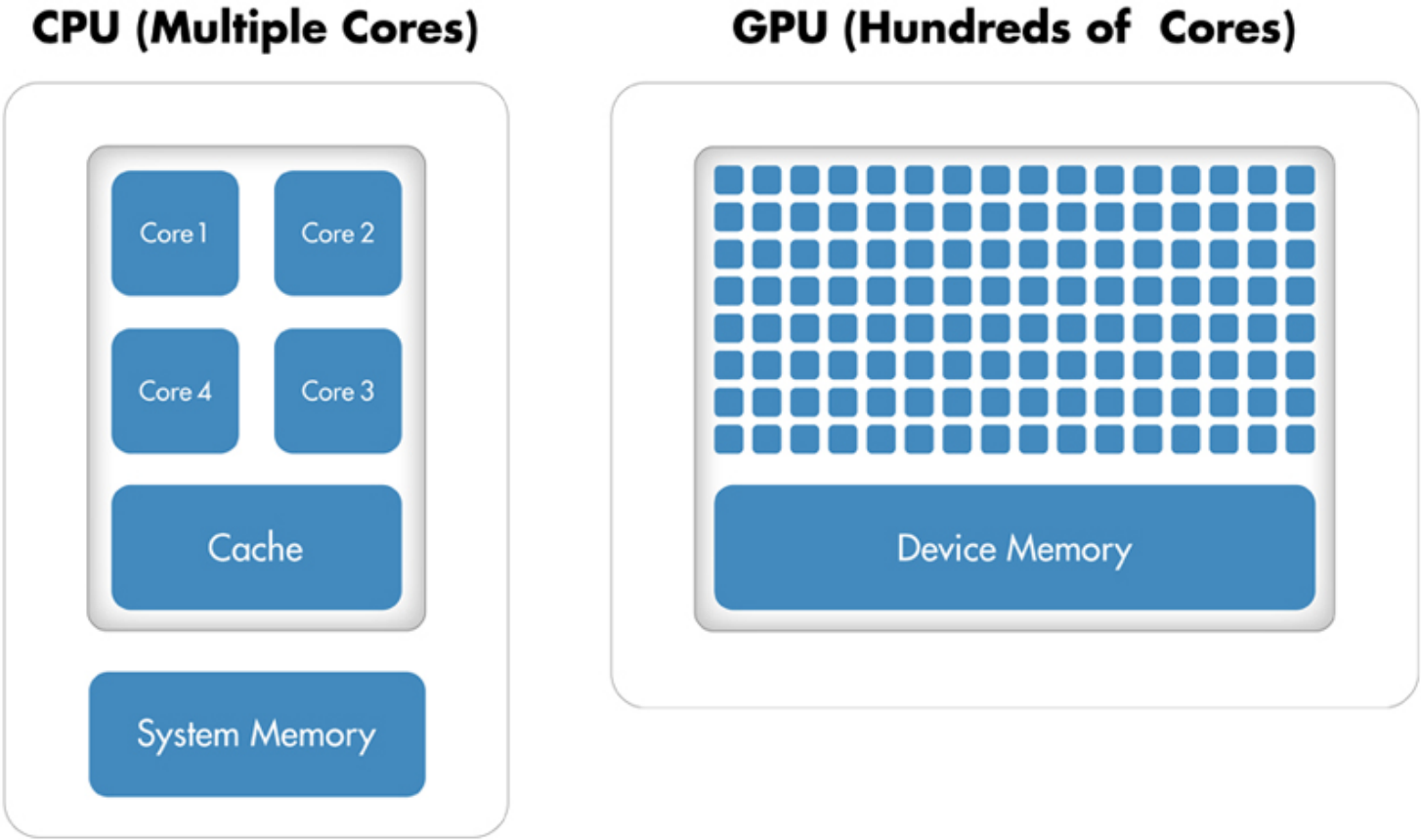
Repeating the Process

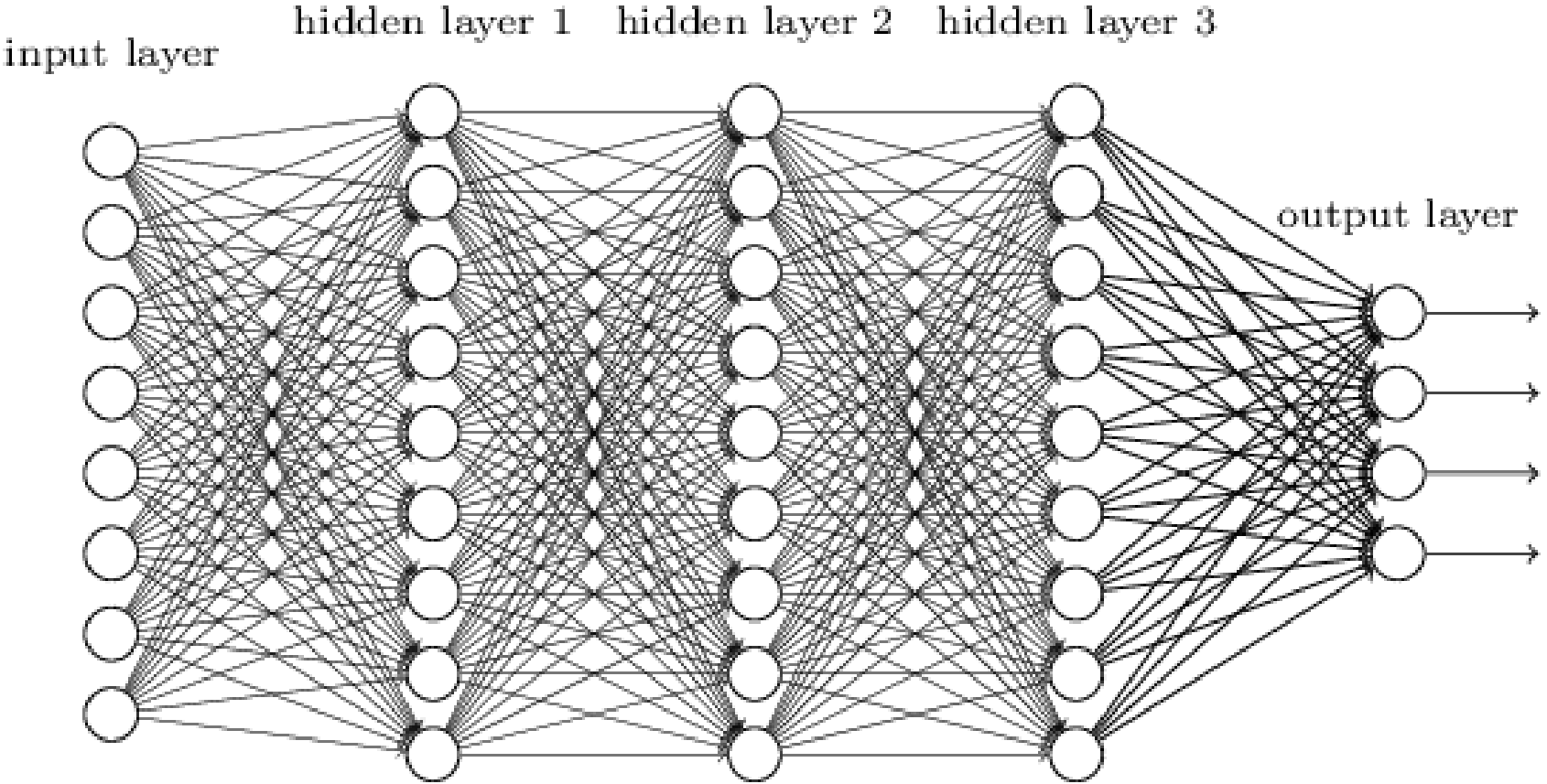
- Convolution -> Max Pooling is repeated some number of times (say 100)
- We can then optionally run the result of that output through another convolution, and another pooling
- The final output is run through a fully connected neural network


Convolution
Pooling
Convolution
Pooling
Fully Connected
Fully Connected
Output Predictions
dog (0.01)
cat (0.04)
boat (0.94)
bird (0.02)
bird
sunset
dog
cat
p_{bird}
p_{sunset}
p_{dog}
p_{cat}
...
vec
convolution + nonlinearity
max pooling
convolution + pooling layers
fully connected layers
Nx binary classification

Computing Required

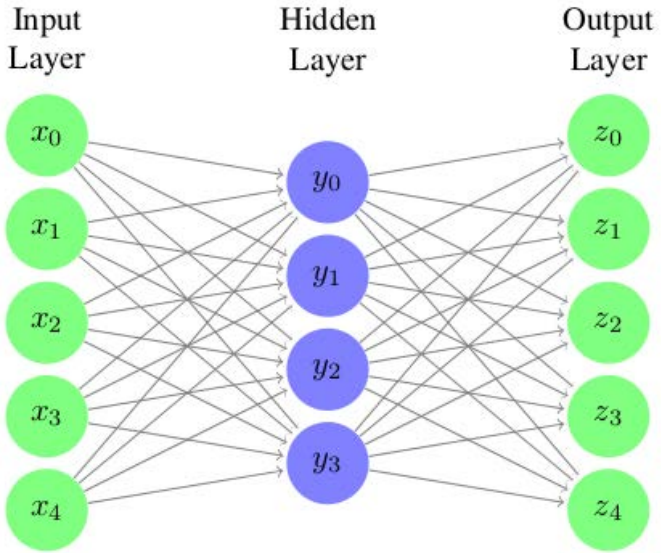
- These techniques are only recently possible due to massive computing power
- GPUs have become ubiquitous and very cheap to manufacture
- Trade single processor speed for massive parallelism
- Neural nets are inherently parallel


CPU (Multiple Cores)
Core 1
Core 2
Core 4
Core 3
Cache
System Memory
GPU (Hundreds of Cores)
Device Memory

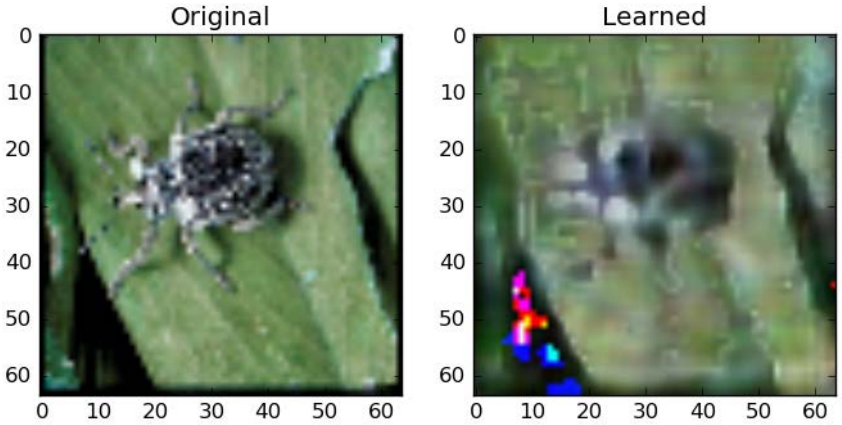

input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer

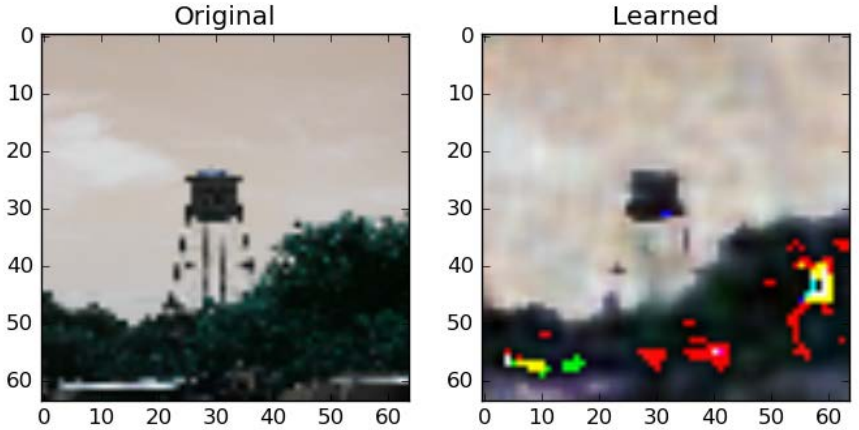
Autoencoding

- Inputs x_1 ... x_n
- Hidden layer which is smaller than input
- Output layer size of input
- Our target values are x_i
- Finding generalizations of patterns on input



Autoencoding Example





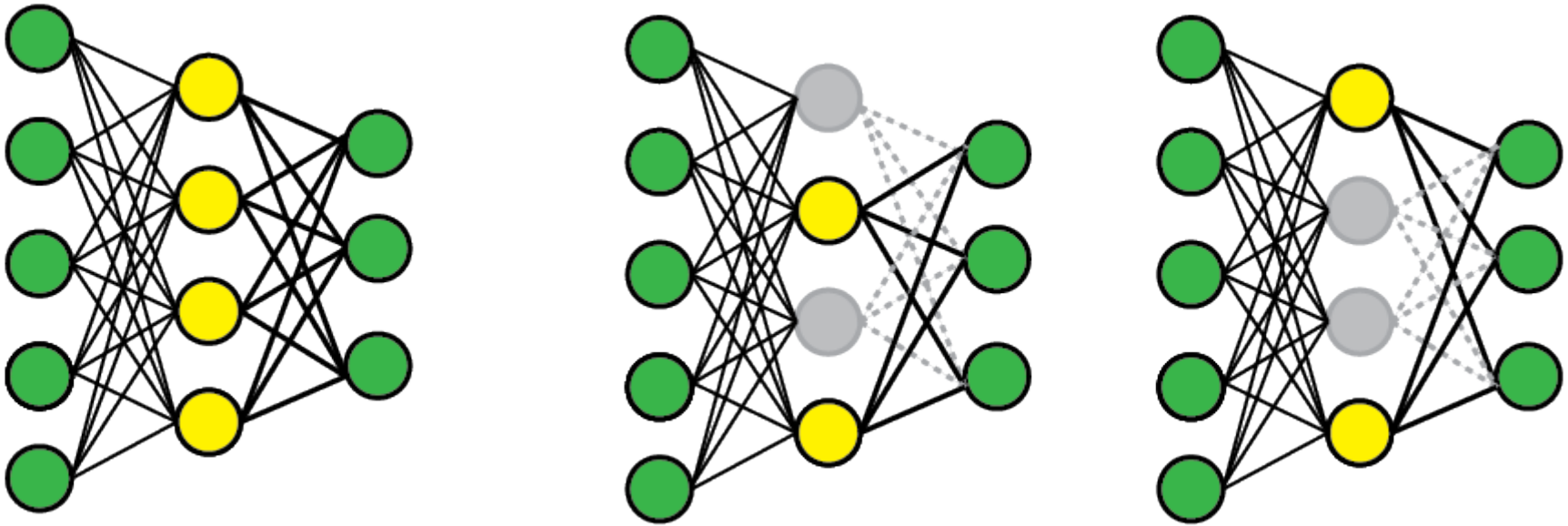
Autoencoding

- Autoencoding “generalizes” inputs
- Can be seen as form of compression
- DNN use autoencoders frequently
- Unlike our demo example, due to these generalizations it is often difficult to understand what the DNN is doing and what features it is extracting

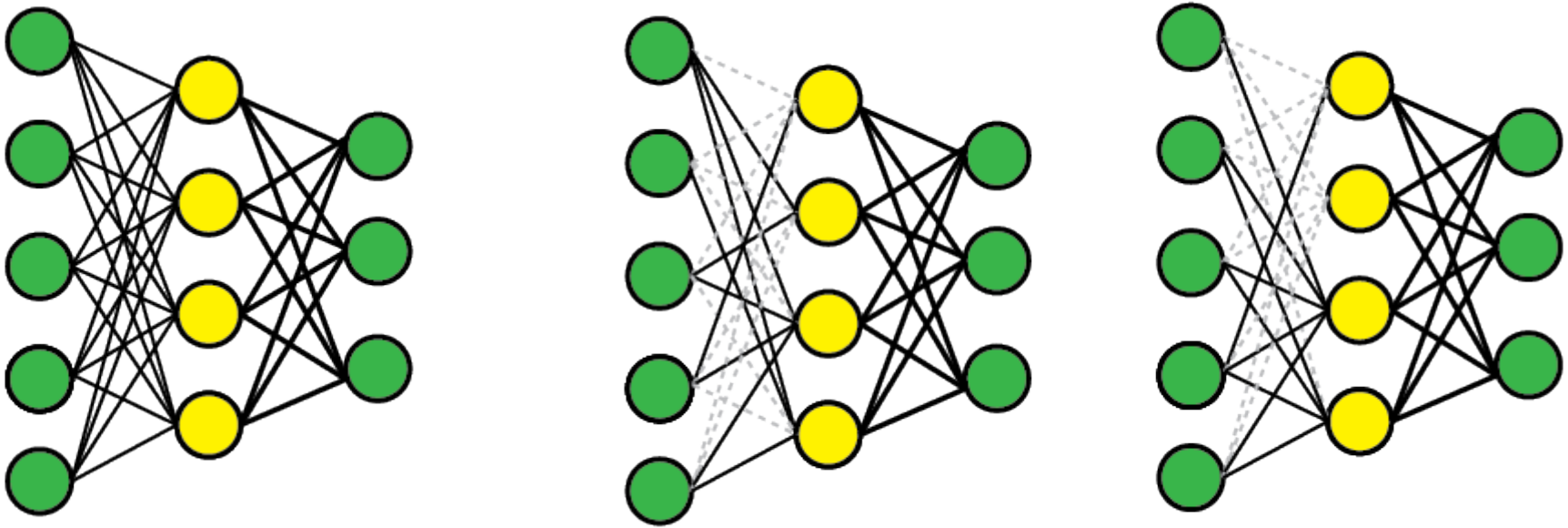
Dropout

- The curse of the neural net is that it can get stuck in a local maxima
- Dropout: If on every iteration we flip a coin for each neuron, and depending on the output we disable the neuron
- Next iteration, drop out a different set
- Prevents getting stuck in local maxima

Dropout

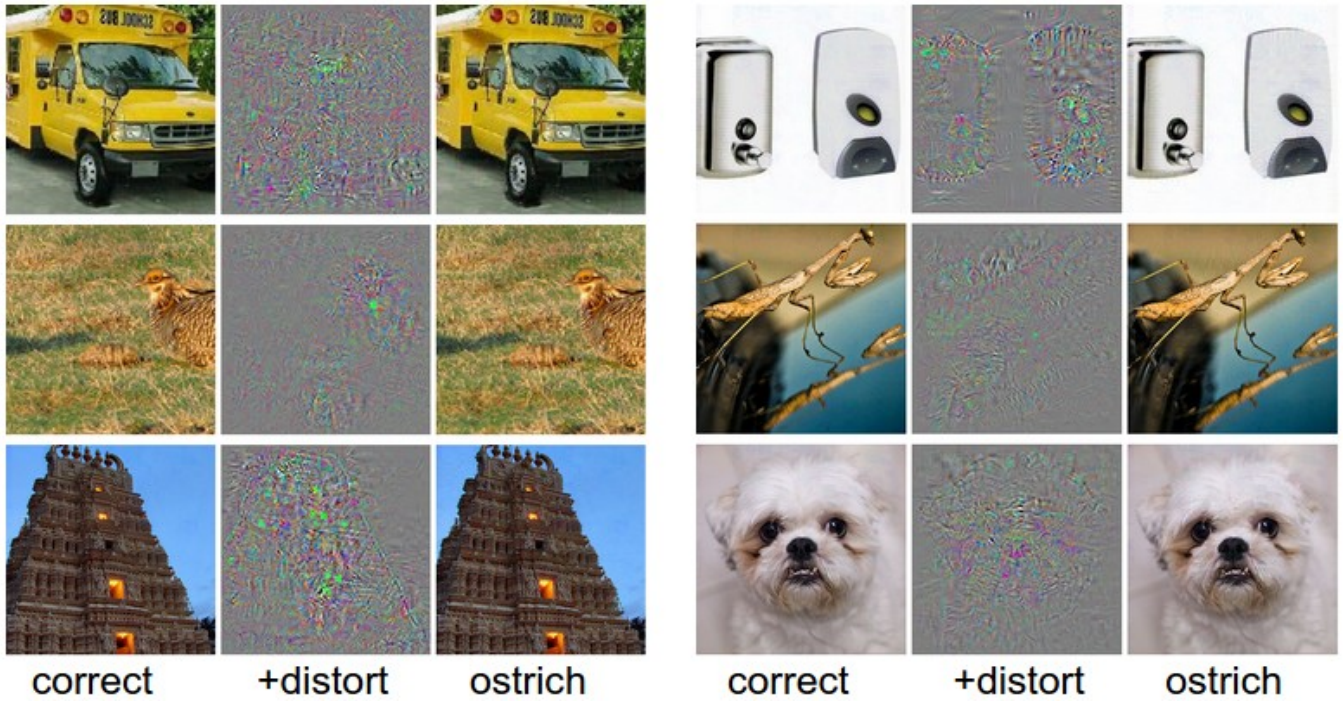


DropConnect (Dropout Generalized)

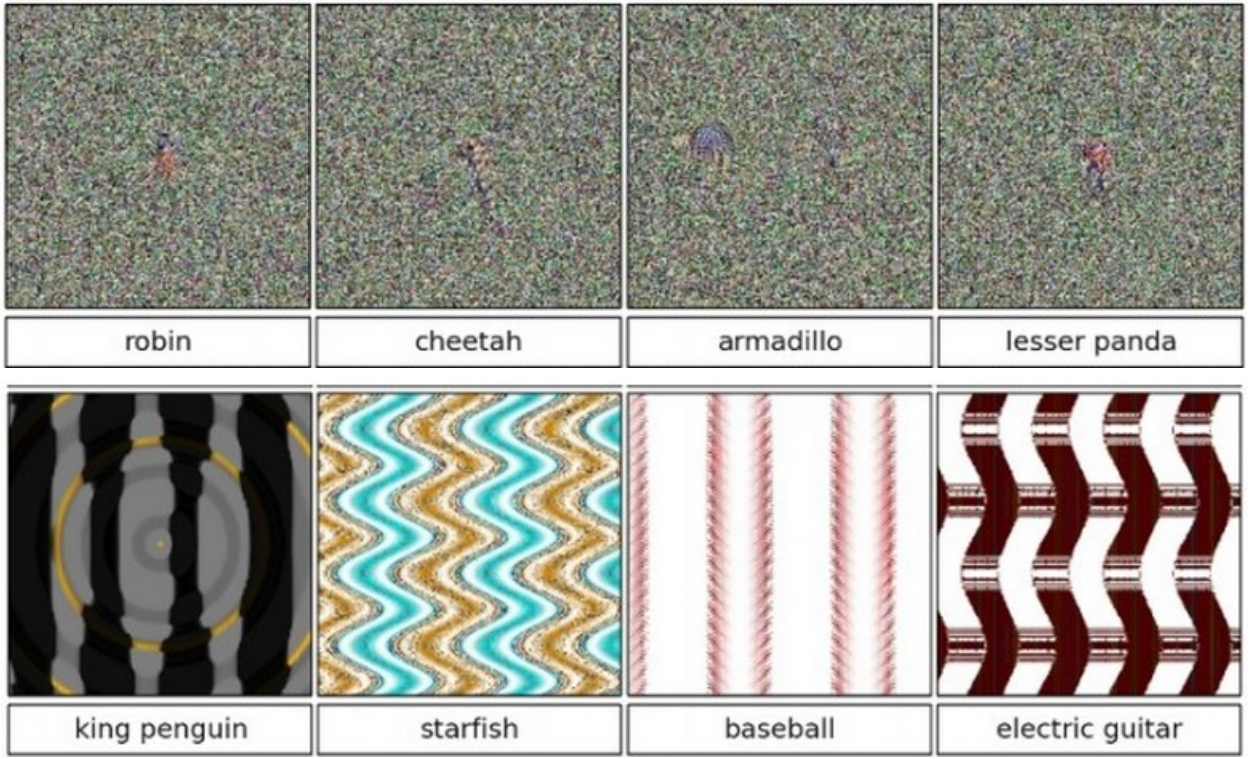


What do DNN see?

Tricking DNN

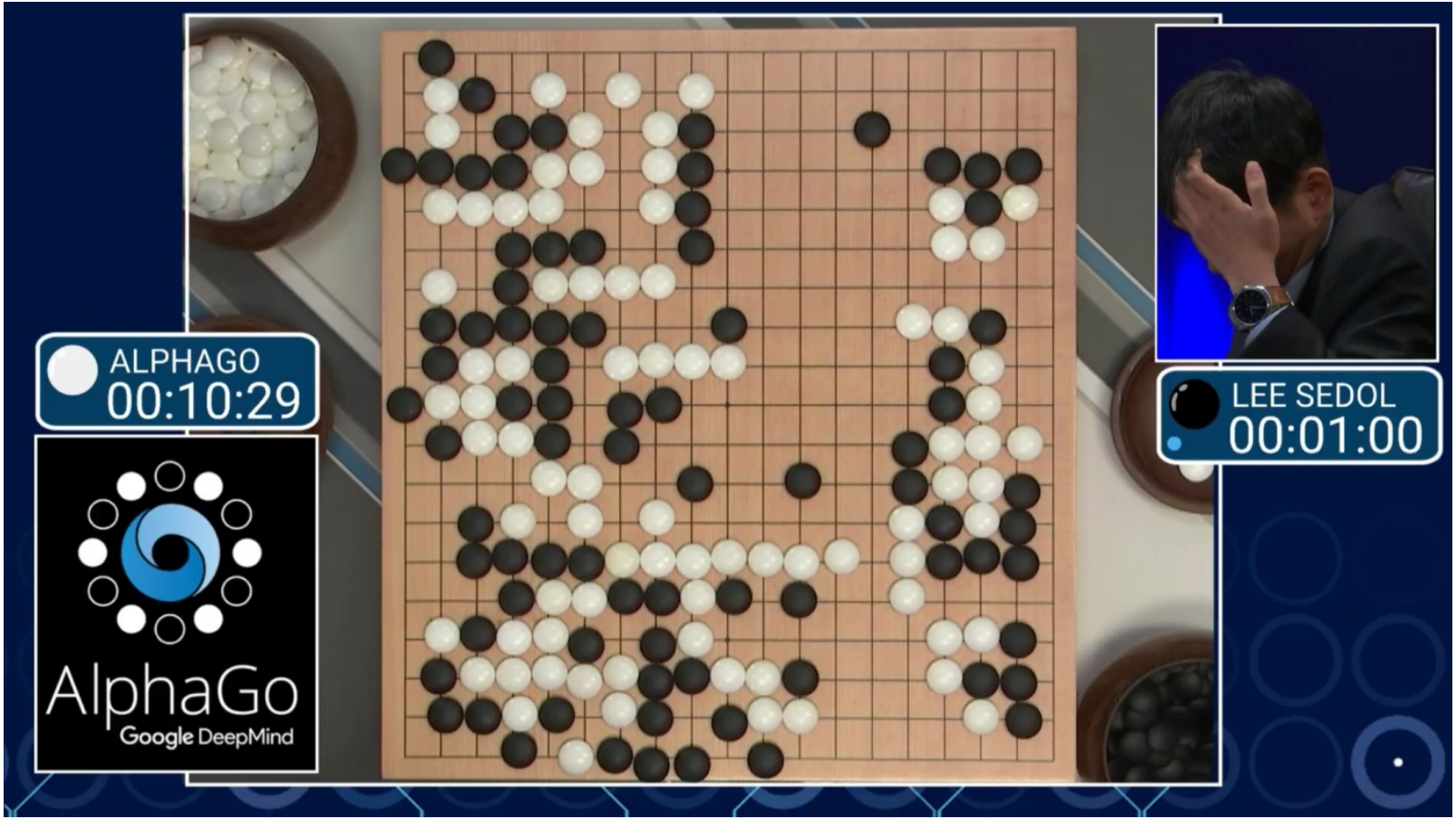


Tricking DNN



Not Human

- Deep Neural Networks are an amazing feat of engineering that produce spectacular results
- They are not magic, they are math
- They are easy to fool, if you try


ALPHAGO
00:10:29
AlphaGo
Google DeepMind
LEE SEDOL
00:01:00

AlphaGo

- Combined Deep Learning with Heuristic Search (MCTS)

- Two Networks were trained:
  - Value Network (how good is state S)
  - Policy Network (what move to do at state S)

Learning from Replays

- DeepMind had thousands of professional human games to learn from
- Policy Network
  - Input = State
  - Target = Move the expert human performed
- Value Network
  - Input = State
  - Target = Who won the game from that state

Learning from Self Play

- Once it had learned all it could from humans, it played against itself
- Continued to learn, and get stronger
- Now it learns from a stronger player each time it plays a new game
- Also key: throw in some random moves to learn about states you may not visit

Heuristic Search

- Go has a large branching factor (300)
- Humans have good abstractions of what moves to perform at a state
- AlphaGo learned a policy network
- Now it can narrow the heuristic search to only consider moves in the policy network
- Search also needs a heuristic function to determine how good a state is (value network)


The Development of Campus Plans