Leibniz Supercomputing Centre

of the Bavarian Academy of Sciences and Humanities

Fundamentals of Deep Learning

Part 1: An Introduction to Deep Learning

PD. Dr. Juan J. Durillo

- Leibniz Supercomputing Centre
- Nvidia DLI Ambassador

Agenda

- Part 1: An Introduction to Deep Learning
- Part 2: How a Neural Network Trains
- Part 3: Convolutional Neural Networks
- Part 4: Data Augmentation and Deployment
- Part 5: Pre-Trained Models
- Part 6: Advanced Architectures

Timings

10:00 – 11:30 Lecture + Hands on

11:30 – 11:45 Coffee Break

11:45 – 13:00 Lecture + Hands on

13:00 – 14:00 Lunch Break

14:00 – 15:30 Lecture + Hands on

15:30 – 15:45 Coffee Break

15:45 – 17:00 Lecture + Hands on

To see lecture notes, make full screen
and click the “notes” button

Welcome!

The Goals of This Course

Get you up and on your feet quickly

Build a foundation to tackle a deep learning project right away

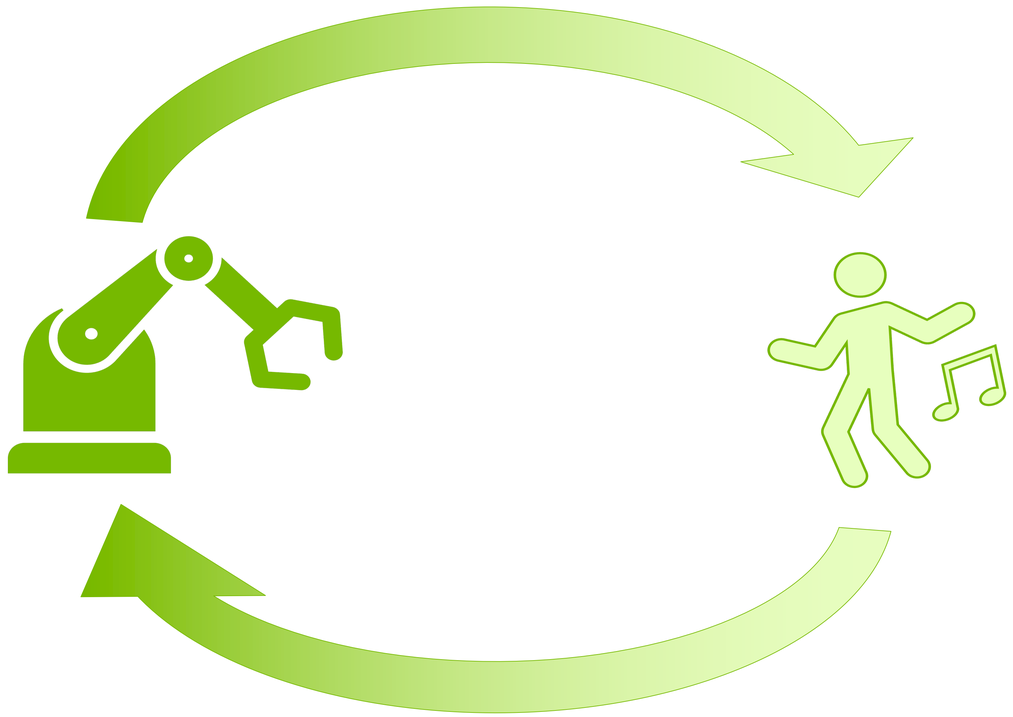
We won’t cover the whole field, but we’ll get a great head start

Foundation from which to read articles, follow tutorials, take further classes

Have Fun!

Human Vs Machine Learning

Relaxed Alertness



Human	Machine
Rest and Digest	Training
Fight-or-flight	Prediction

Let’s Get Started

History of AI

Beginning of Artificial intelligence

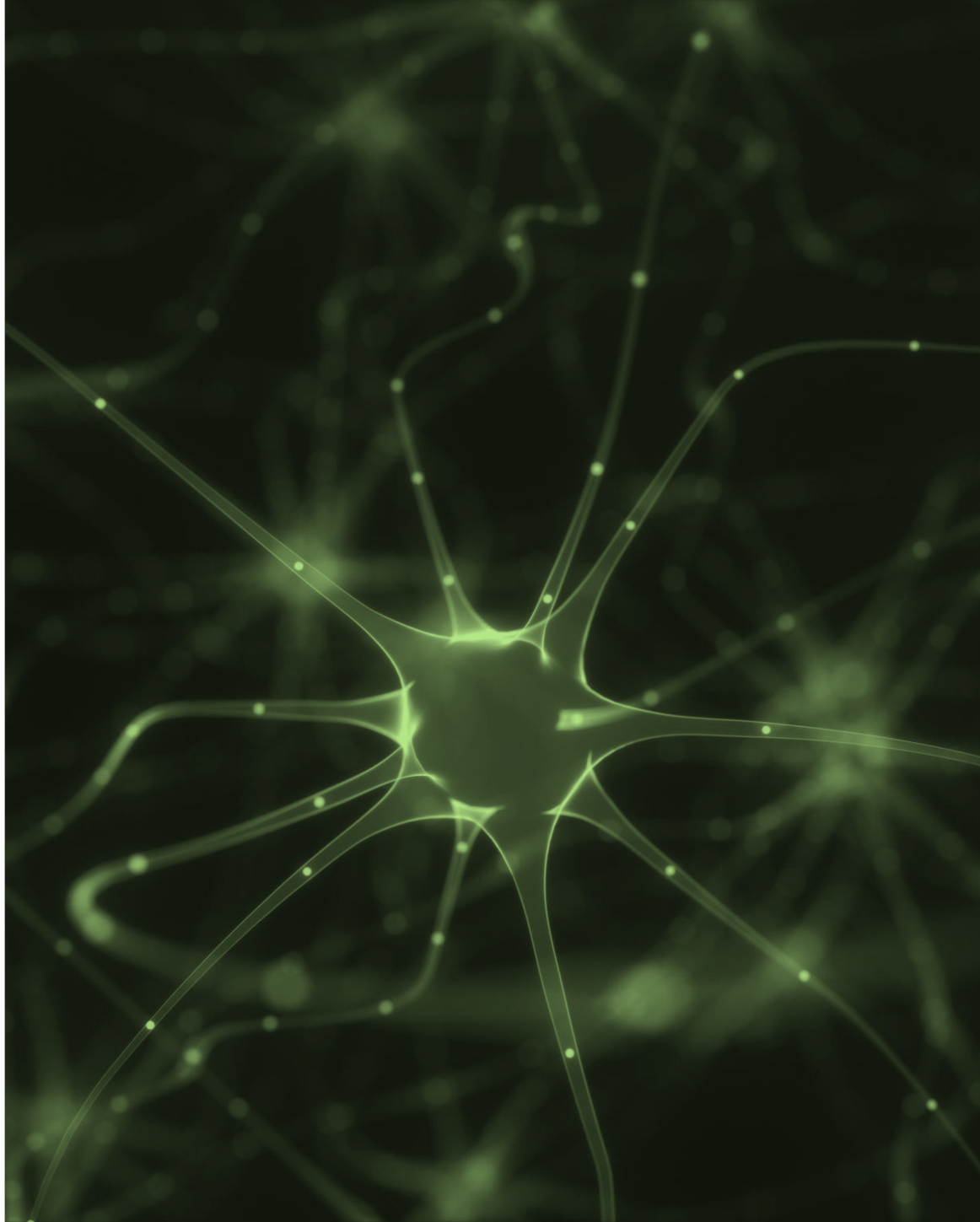
Computers are made in part to complete human tasks

Early on, generalized intelligence looked possible

Turned out to be harder than expected

Early Neural Networks

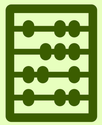
- Inspired by biology
- Created in the 1950’s
- Outclassed by Von Neumann Architecture



Expert Systems

Highly complex

Programmed by hundreds of engineers



Rigorous programming of many rules

Expert Systems - Limitations

What are these three images?

How Do Children Learn?

- Expose them to lots of data
- Give them the “correct answer”
- They will pick up the important patterns on their own

The Deep Learning Revolution

Data

- Networks need a lot of information to learn from
- The digital era and the internet has supplied that data

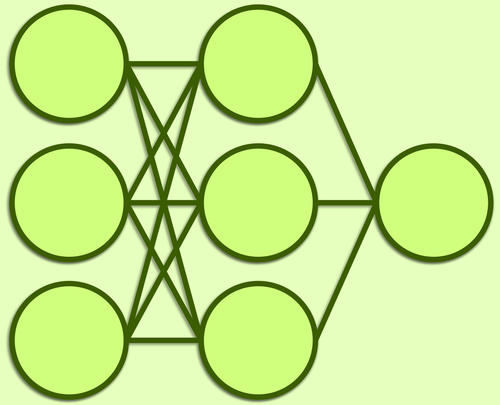
Computing Power

Need a way for our artificial “brain” to observe lots of data within a practical amount of time.

The Importance of the GPU

A Rendered Image

A Neural Network



What is Deep Learning?

“

Deep learning flips traditional programming on its head

”

Traditional Programming

Building a Classifier

1. Define a set of rules for classification
2. Program those rules into the computer
3. Feed it examples, and the program uses the rules to classify

Machine Learning

Building a Classifier

1. Show model the examples with the answer of how to classify
2. Model takes guesses, we tell it if it’s right or not
3. Model learns to correctly categorize as it’s training. The system learns the rules on its own

This is a Fundamental Shift

When to Choose Deep Learning

Classic Programming

If rules are clear and straightforward, often better to program it

Deep Learning

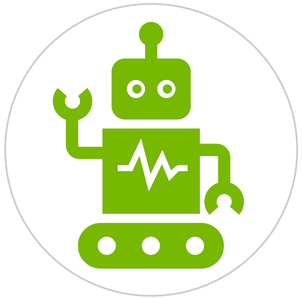
If rules are nuanced, complex, difficult to discern, use deep learning

Deep Learning Compared to Other AI

- Depth and complexity of networks
- Up to billions of parameters (and growing)
- Many layers in a model
- Important for learning complex rules

How Deep Learning is Transforming the World

Computer Vision



ROBOTICS AND
MANUFACTURING

OBJECT DETECTION

SELF-DRIVING CARS

Natural Language Processing

REAL-TIME
TRANSLATION

VOICE
RECOGNITION

VIRTUAL ASSISTANTS

Recommender Systems

CONTENT CURATION

TARGETED ADVERTISING

SHOPPING RECOMMENDATIONS

Reinforcement Learning

ALPHAGO BEATS
WORLD CHAMPION
IN GO

AI BOTS BEAT
PROFESSIONAL
VIDEOGAMERS

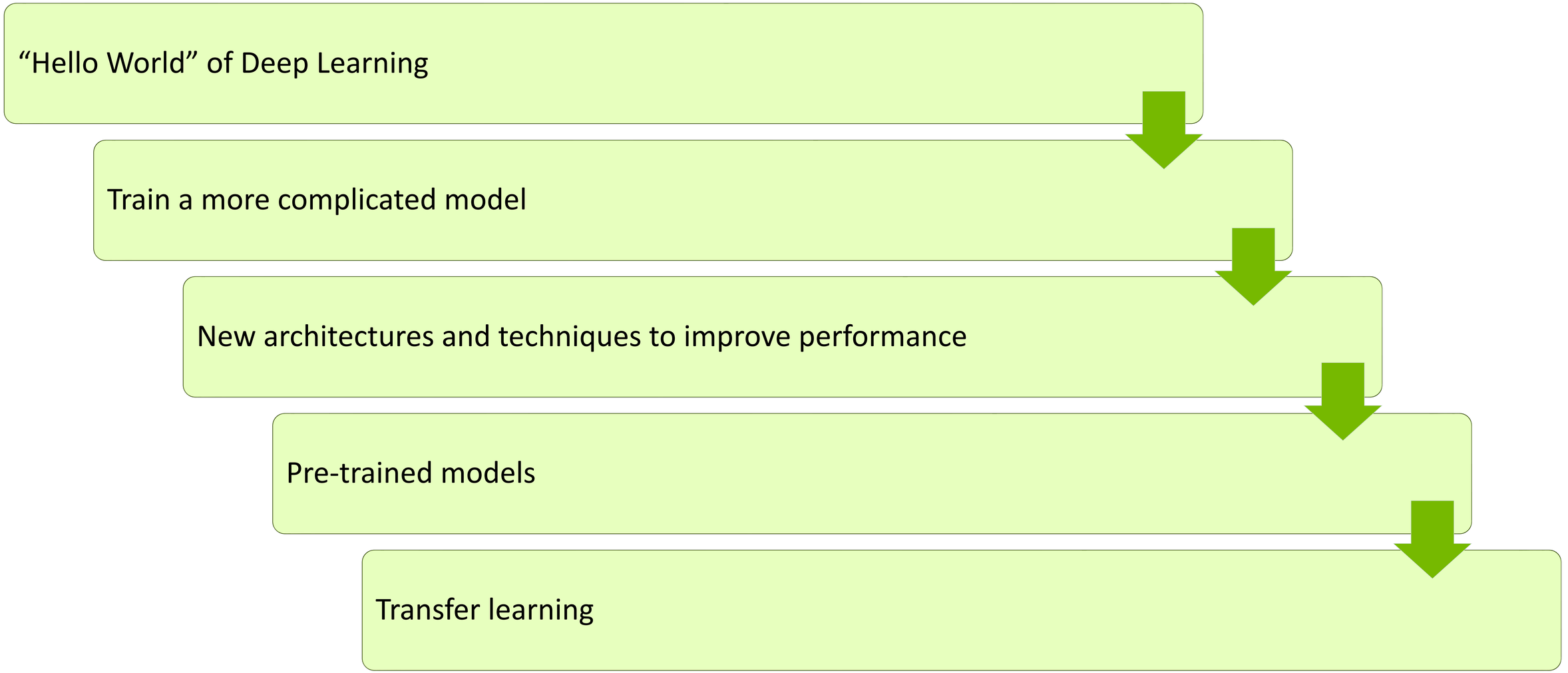
STOCK TRADING
ROBOTS

Overview of the Course

Hands on Exercises

- Get comfortable with the process of deep learning
- Exposure to different models and datatypes
- Get a jump-start to tackle your own projects

Structure of the Course



Platform of the Course

GPU powered cloud server

JupyterLab platform

Jupyter notebooks for interactive coding

Software of This Course

- Major deep learning platforms:
  - TensorFlow + Keras (Google)
  - PyTorch (Meta)
  - MXNet (Apache)
- We'll be using PyTorch
- Good idea to gain exposure to others moving forward

Hello Neural Networks

Train a network to correctly classify handwritten digits

Historically important and difficult task for computers

Try learning like a Neural Network

Get exposed to the example, and try to figure out the rules to how it works

Let’s Go!

Fundamentals of Deep Learning

Part 2: How a Neural Network Trains

Agenda

- Part 1: An Introduction to Deep Learning
- Part 2: How a Neural Network Trains
- Part 3: Convolutional Neural Networks
- Part 4: Data Augmentation and Deployment
- Part 5: Pre-Trained Models
- Part 6: Advanced Architectures

Recap of the Exercise

What just happened?

Loaded and visualized our data

Edited our data (reshaped, normalized, to categorical)

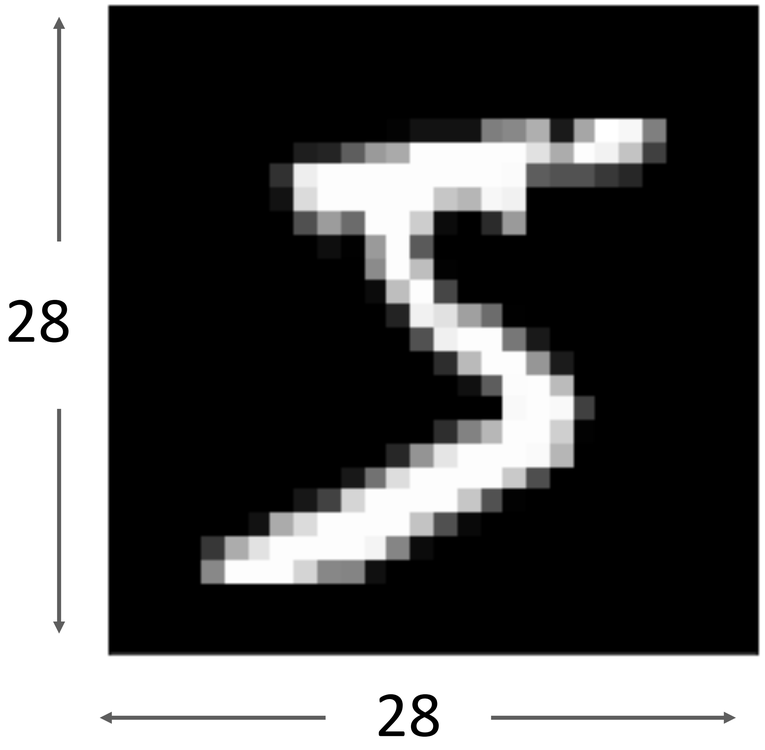
Created our model

Compiled our model

Trained the model on our data

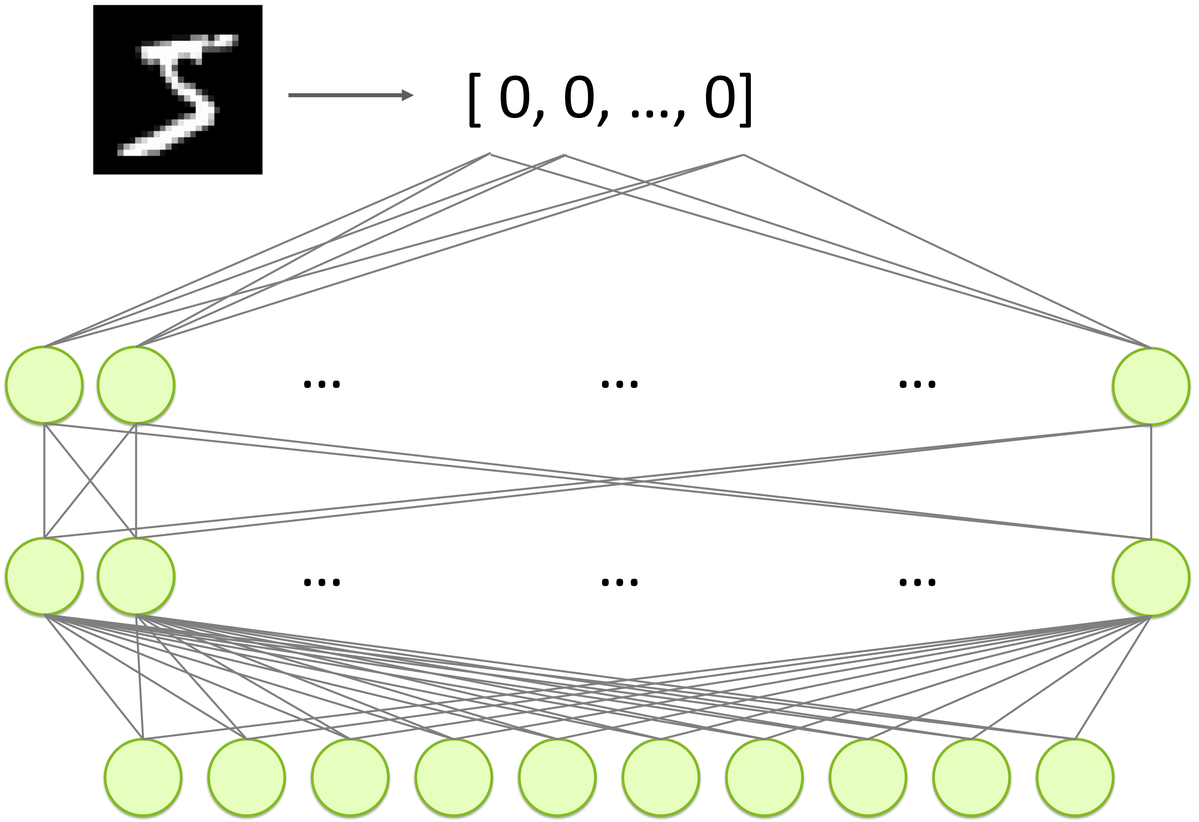
Data Preparation

Input as an Array



[0,0,0,24,75,184,185,78,32,55,0,0,0…]

An Untrained Model

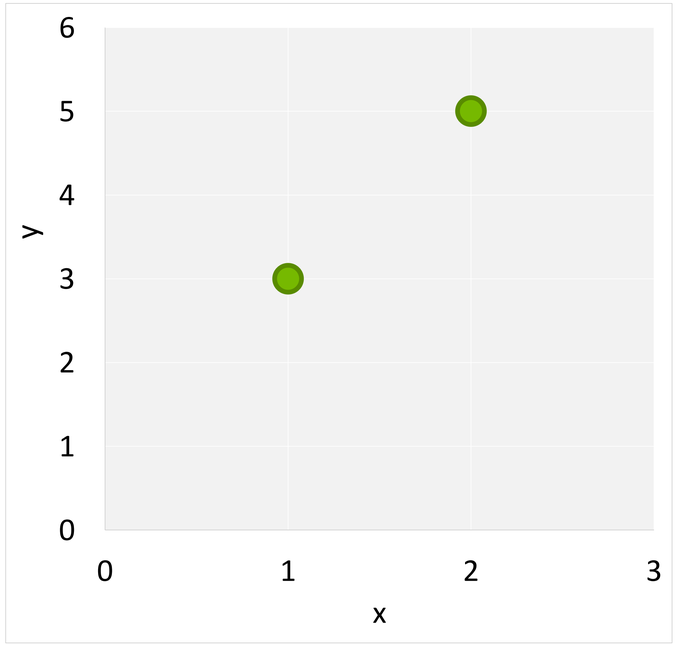


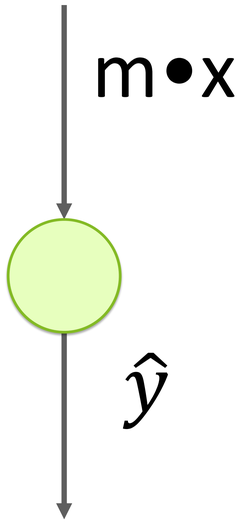
A Simpler Model

A Simpler Model

$$y = mx + b$$

x	y
1	3
2	5





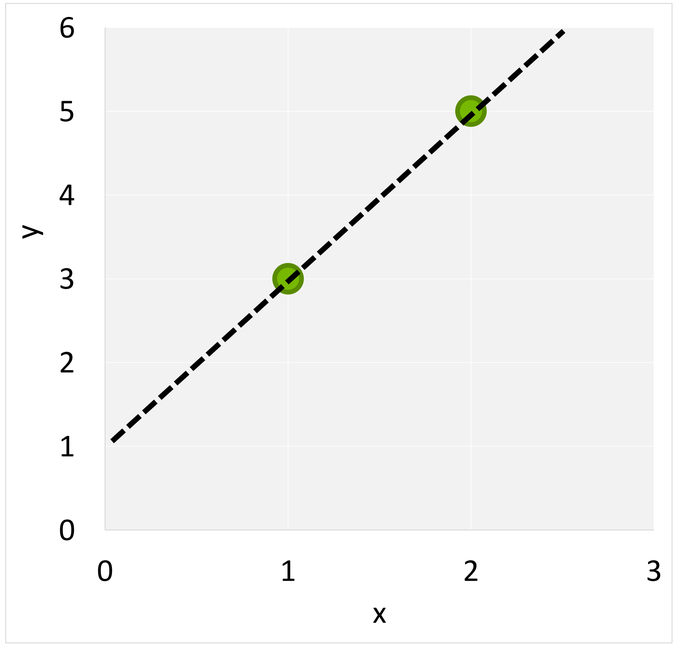
$m = ?$

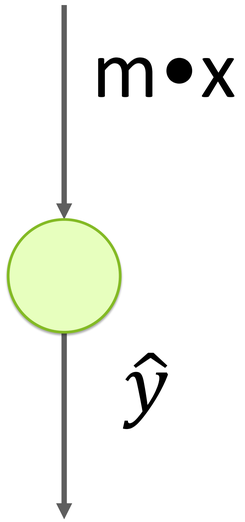
$b = ?$

A Simpler Model

$$y = mx + b$$

x	y
1	3
2	5





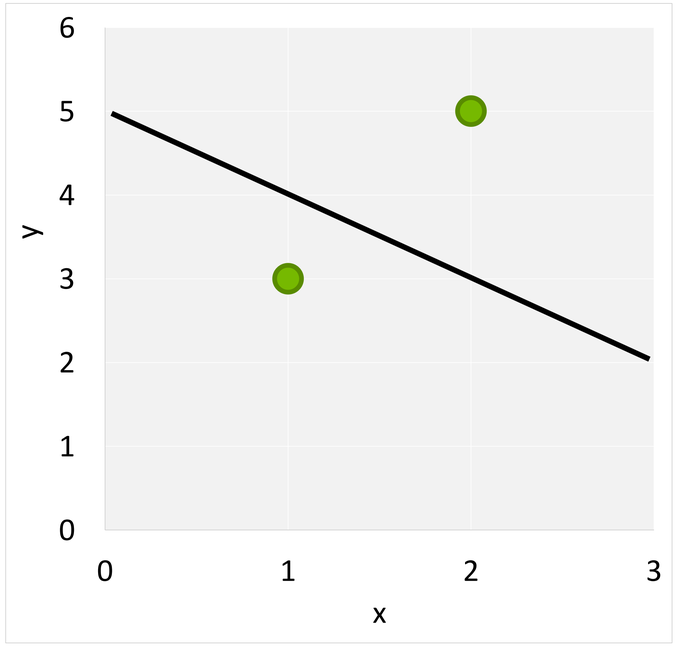
$m = ?$

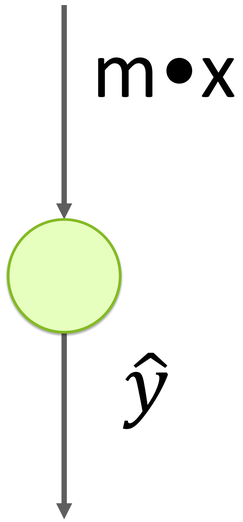
$b = ?$

A Simpler Model

$$y = mx + b$$

x	y	$\hat{y}$
1	3	4
2	5	3





Start Random

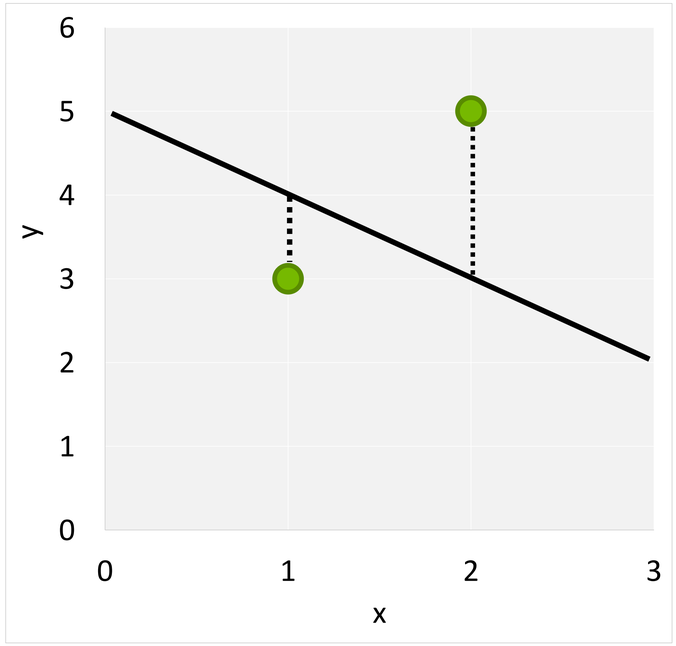
$m = -1$

$b = 5$

A Simpler Model

$$y = mx + b$$

x	y	$\hat{y}$	err^2
1	3	4	1
2	5	3	4
		MSE =	2.5
		RMSE =	1.6



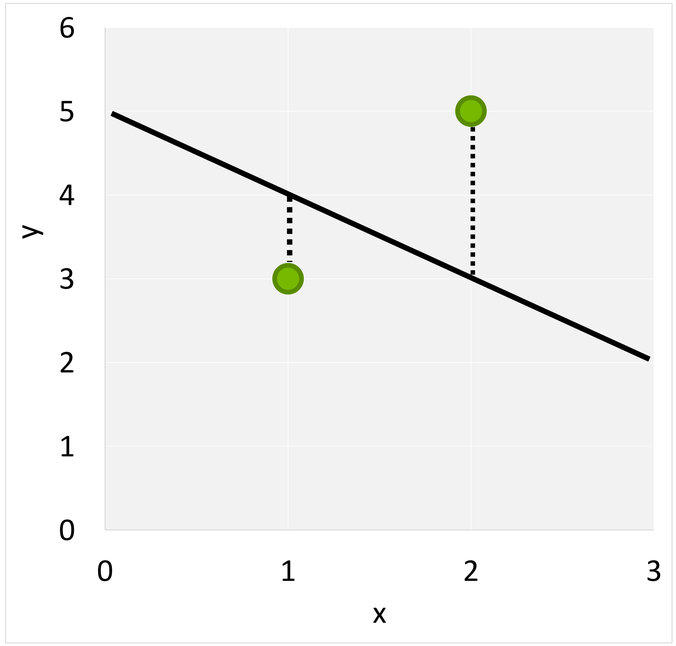
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

A Simpler Model

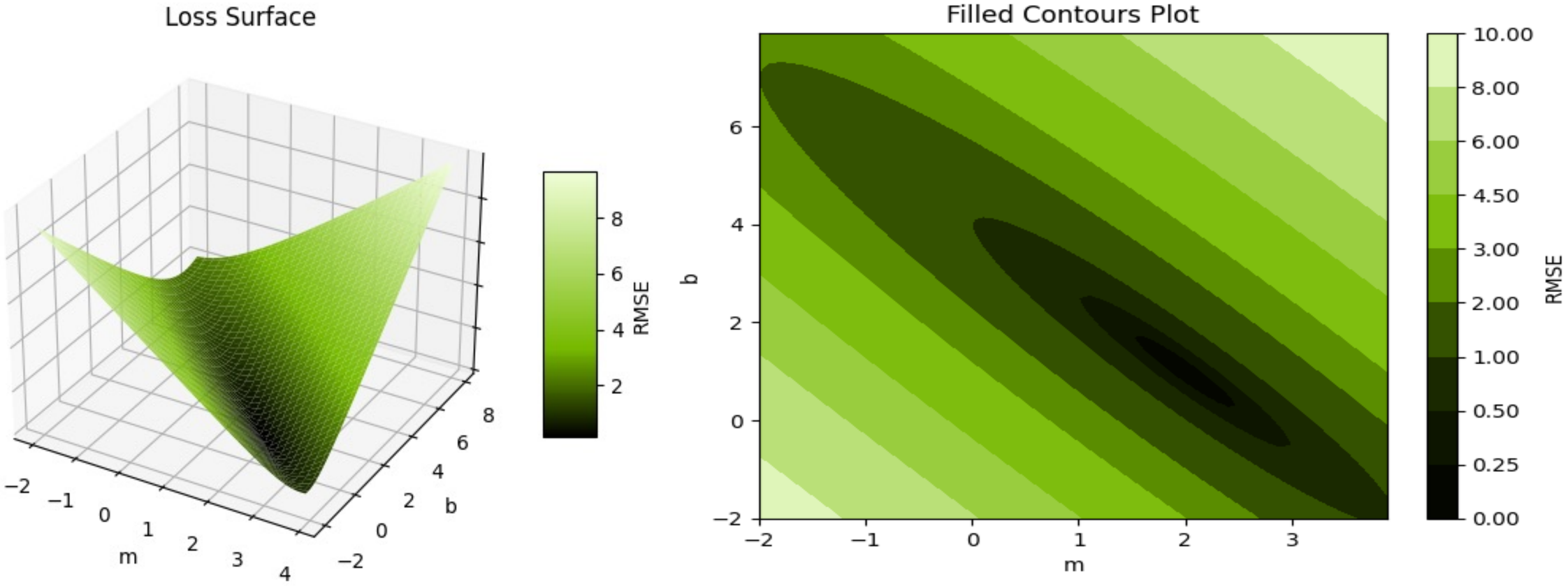
$$y = mx + b$$

x	y	$\hat{y}$	err^2
1	3	4	1
2	5	3	4
		MSE =	2.5
		RMSE =	1.6

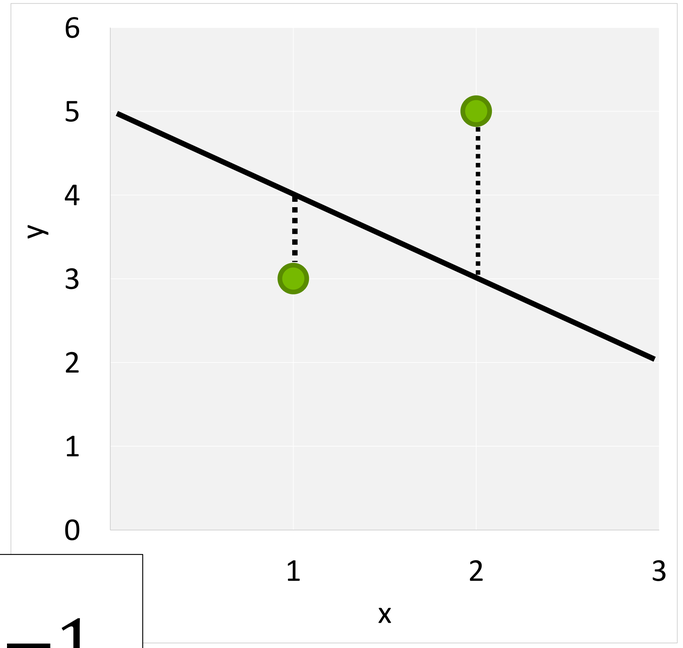



```python
data = [(1, 3), (2, 5)]
m = -1
b = 5


def get_rmse(data, m, b):
    """Calculates Mean Square Error"""
    n = len(data)
    squared_error = 0
    for x, y in data:
        # Find predicted y
        y_hat = m*x+b
        # Square difference between
        # prediction and true value
        squared_error += (
            y - y_hat)**2
    # Get average squared difference
    mse = squared_error / n
    # Square root for original units
    return mse ** .5

```

The Loss Curve

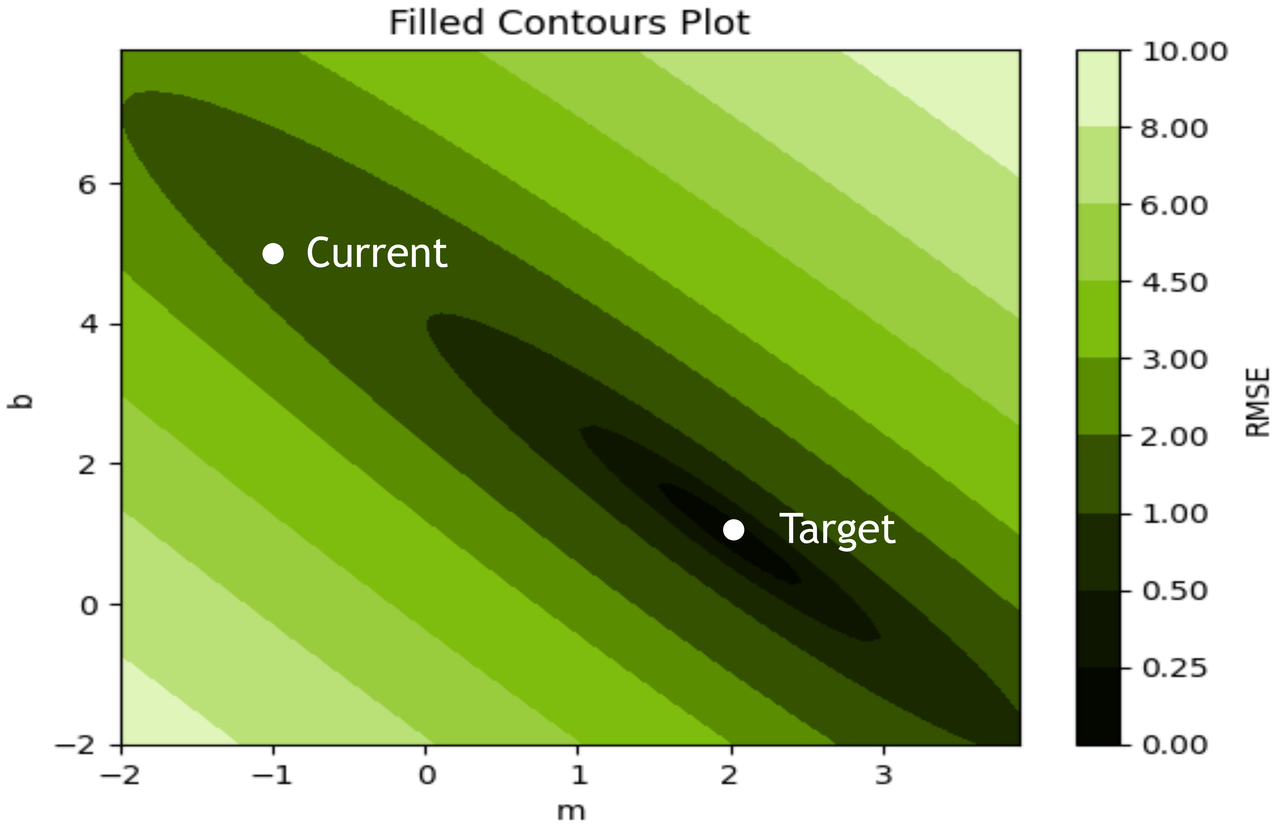


The Loss Curve

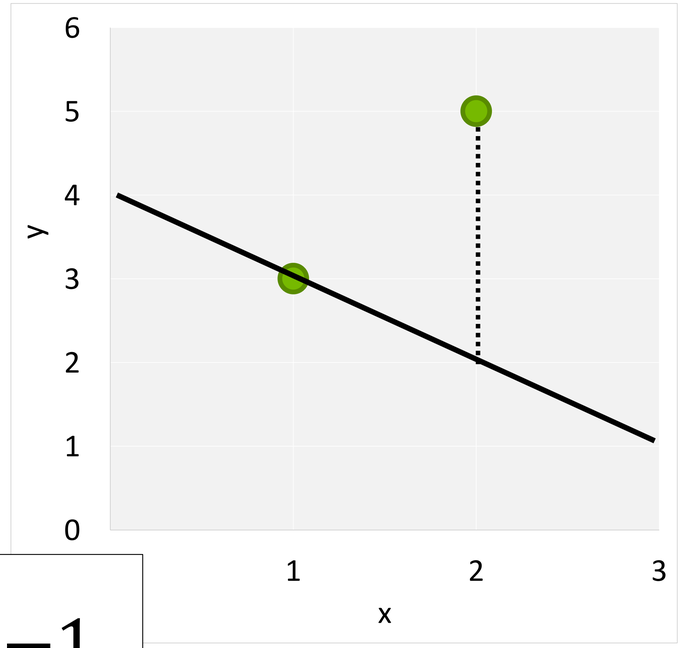


$m = -1$

$b = 5$

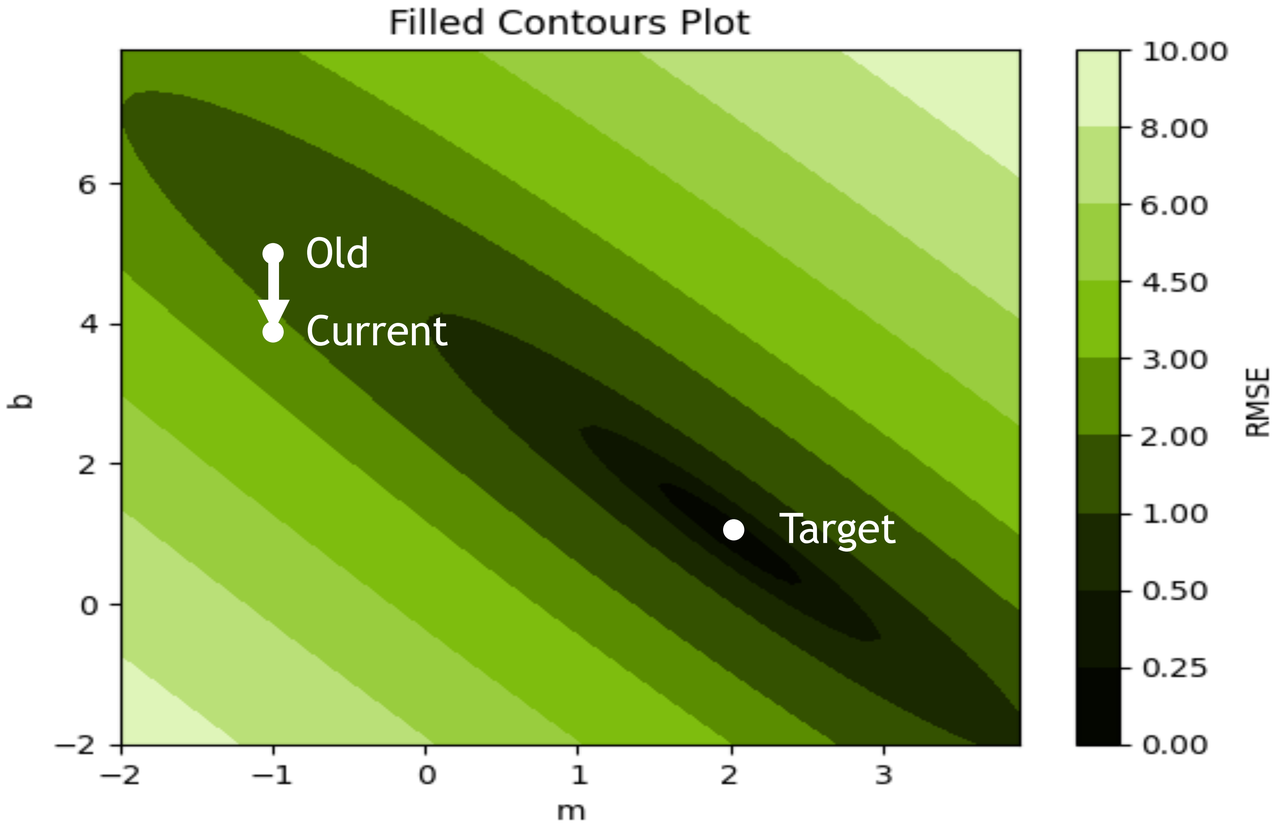


The Loss Curve

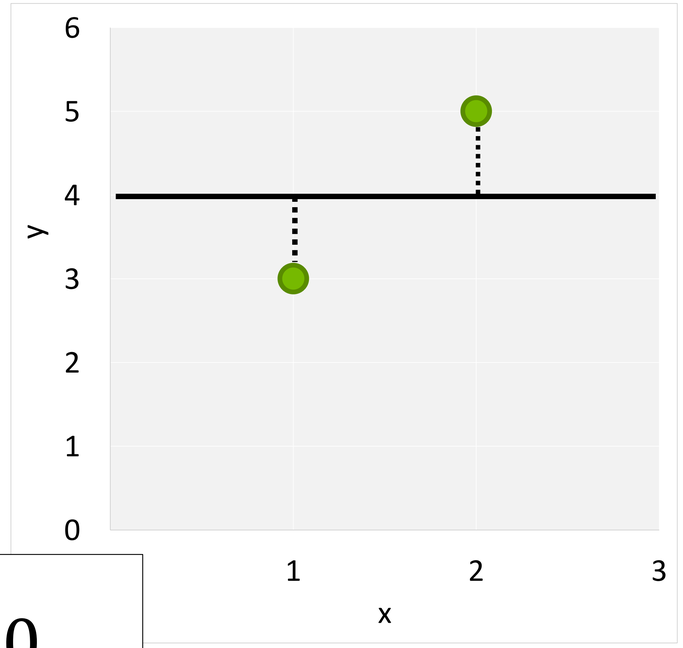


$m = -1$

$\mathrm{b} = 4$

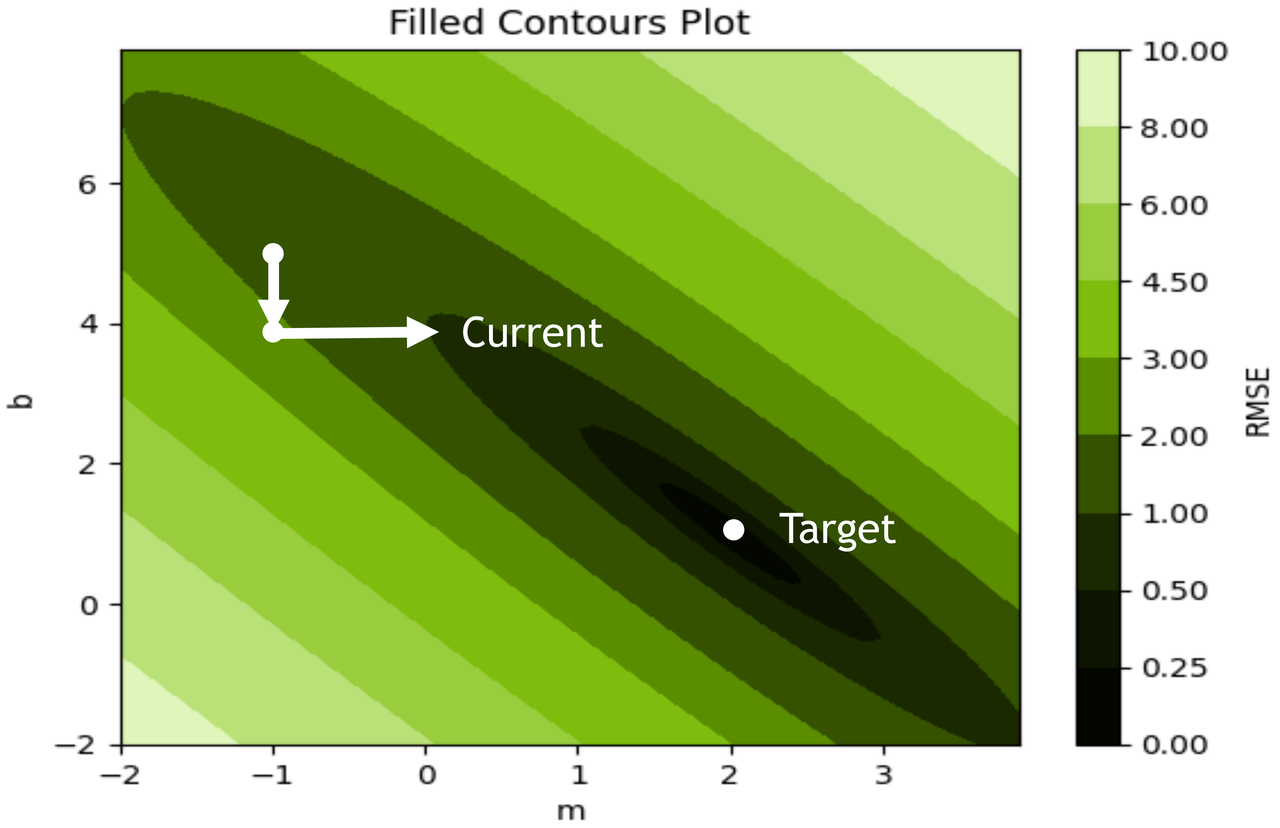


The Loss Curve



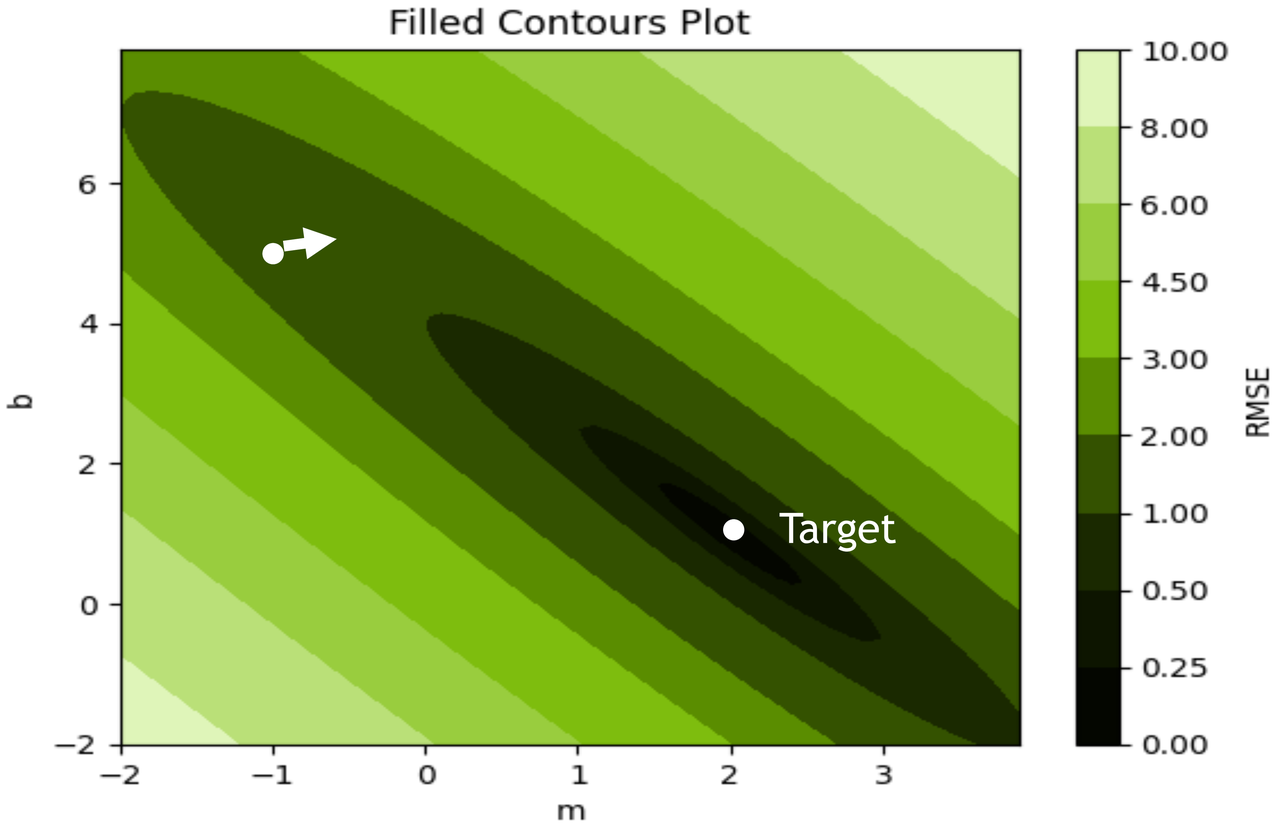
$m = 0$

$\mathrm{b} = 4$



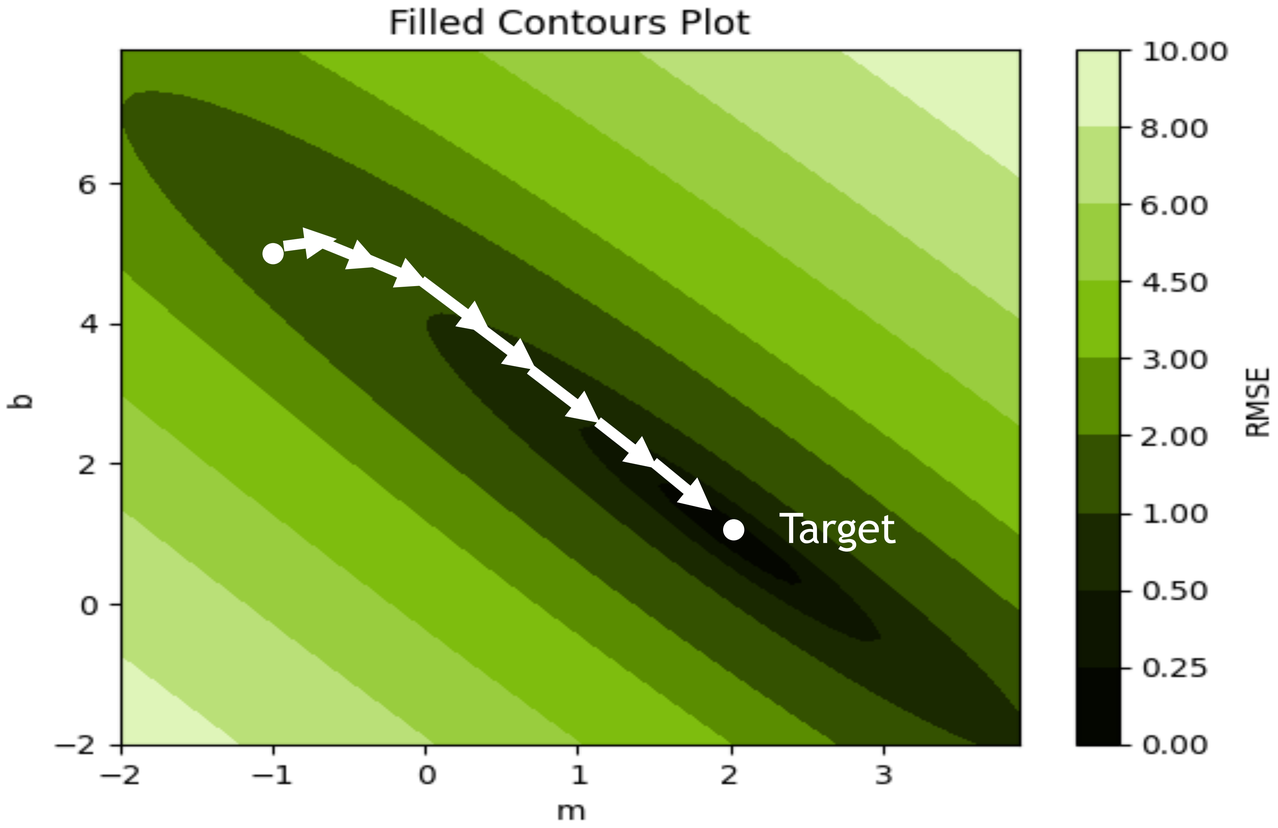
The Loss Curve

The Gradient	Which direction loss decreases the most
λ: The learning rate	How far to travel
Epoch	A model update with the full dataset
Batch	A sample of the full dataset
Step	An update to the weight parameters

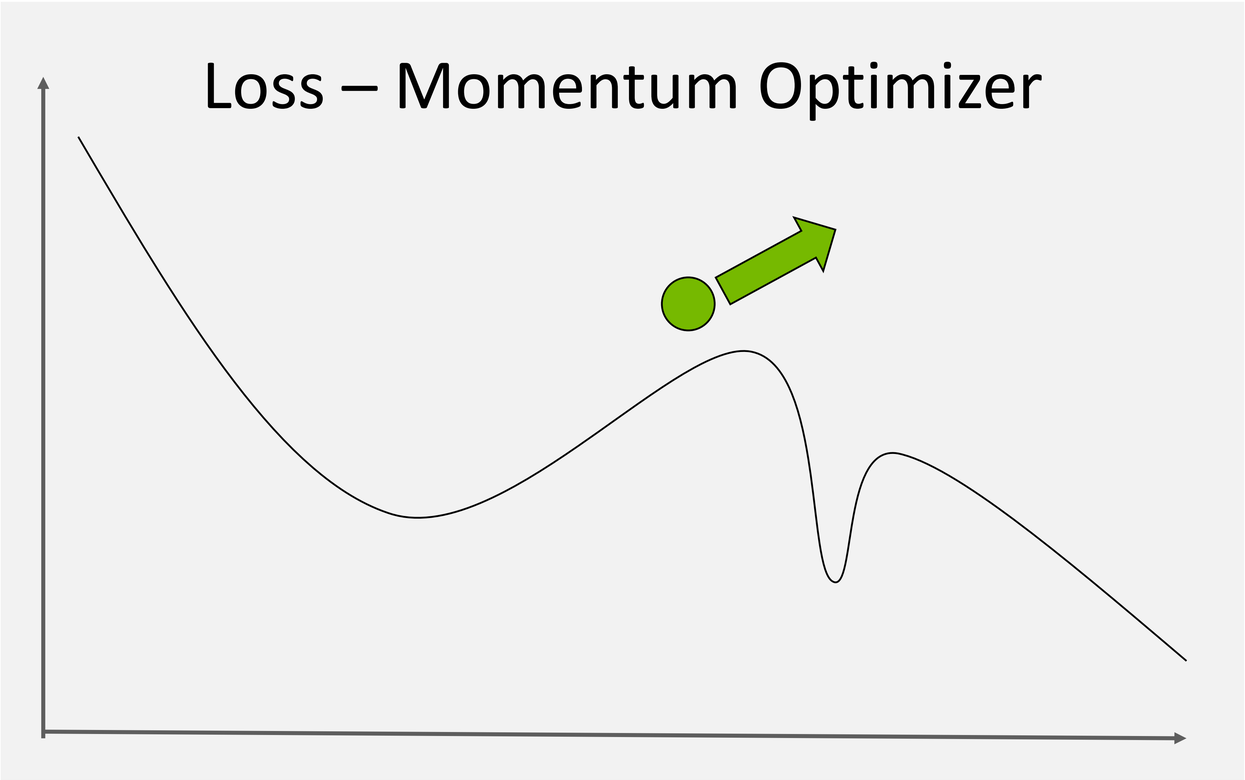


The Loss Curve

The Gradient	Which direction loss decreases the most
λ: The learning rate	How far to travel
Epoch	A model update with the full dataset
Batch	A sample of the full dataset
Step	An update to the weight parameters



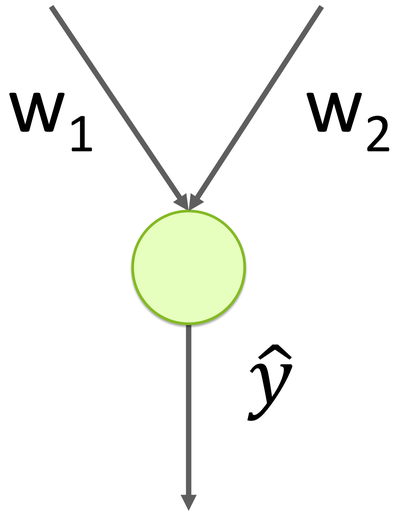
Optimizers



- Adam
- Adagrad
- RMSprop
- SGD

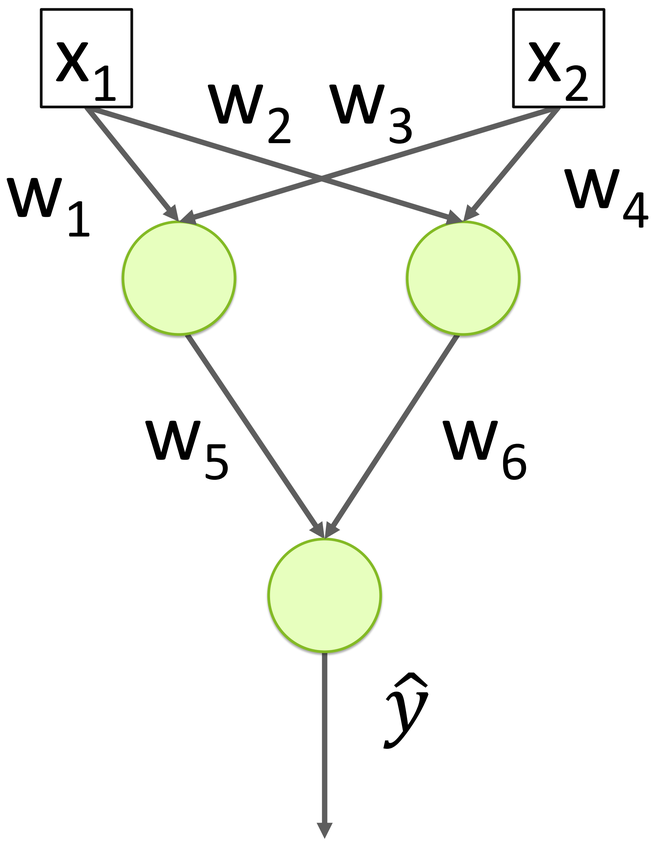
From Neuron to Network

Building a Network



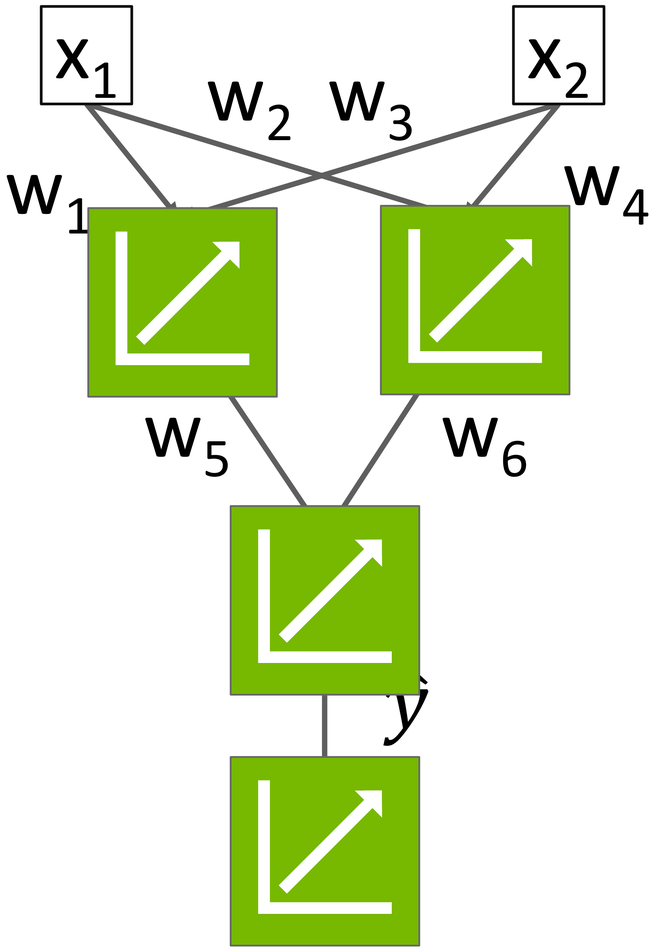
- Scales to more inputs

Building a Network



- Scales to more inputs
- Can chain neurons

Building a Network



- Scales to more inputs
- Can chain neurons
- If all regressions are linear, then output will also be a linear regression

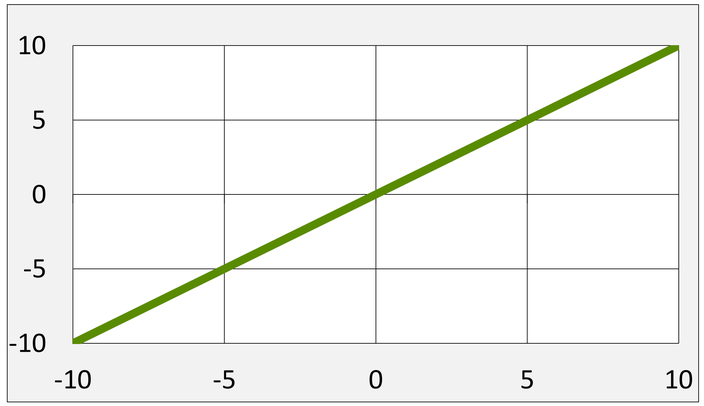
Activation Functions

Activation Functions

Linear

$$\hat{y} = wx + b$$

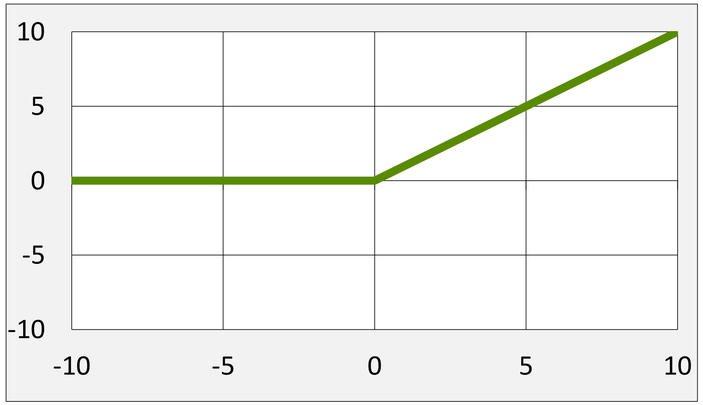
```
# Multiply each input
# with a weight (w) and
# add intercept (b)
y_hat = wx+b
```



ReLU

$$\hat{y} = \begin{cases} wx + b \; if \; wx + b > 0 \\ 0 \; otherwise \end{cases}$$

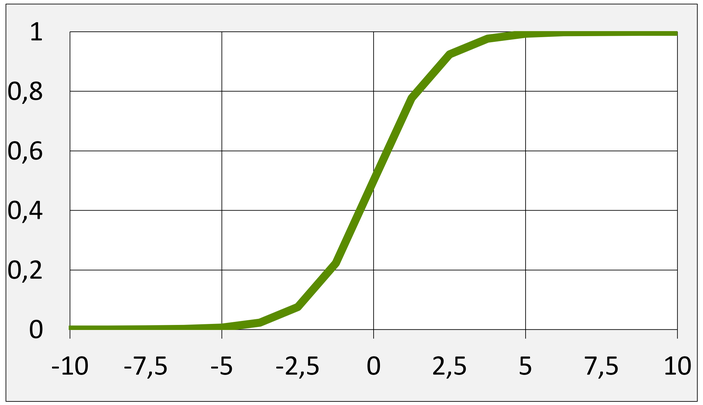
```
# Only return result
# if total is positive
linear = wx+b
y_hat = linear * (linear > 0)
```



Sigmoid

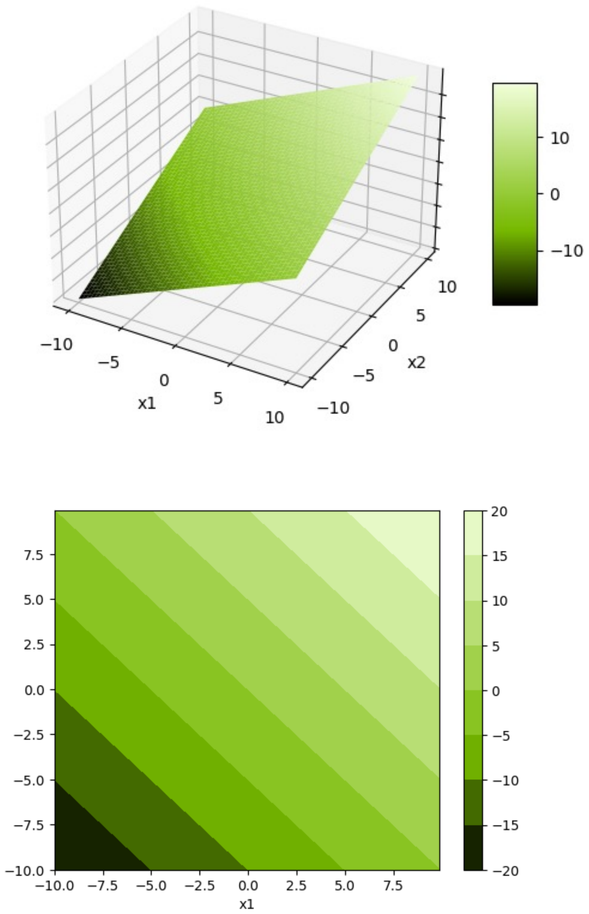
$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$

```
# Start with line
linear = wx + b
# Warp to - inf to 0
inf_to_zero = np.exp(-1 * linear)
# Squish to -1 to 1
y_hat = 1 / (1 + inf_to_zero)
```

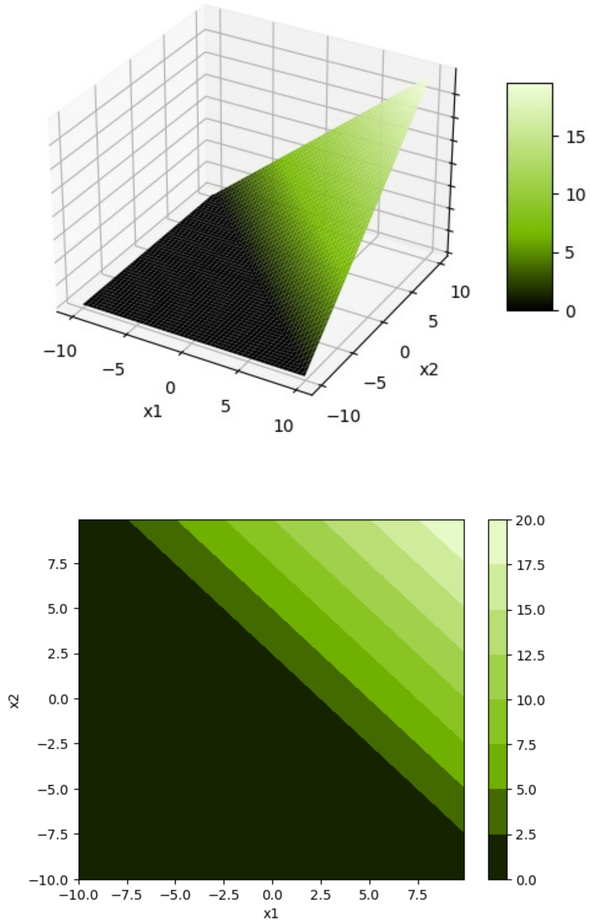


Activation Functions

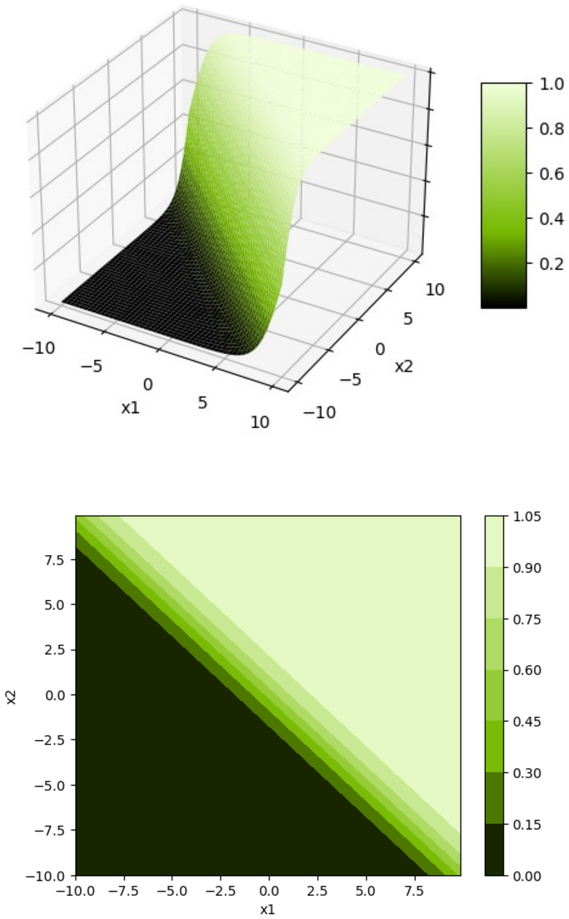
Linear



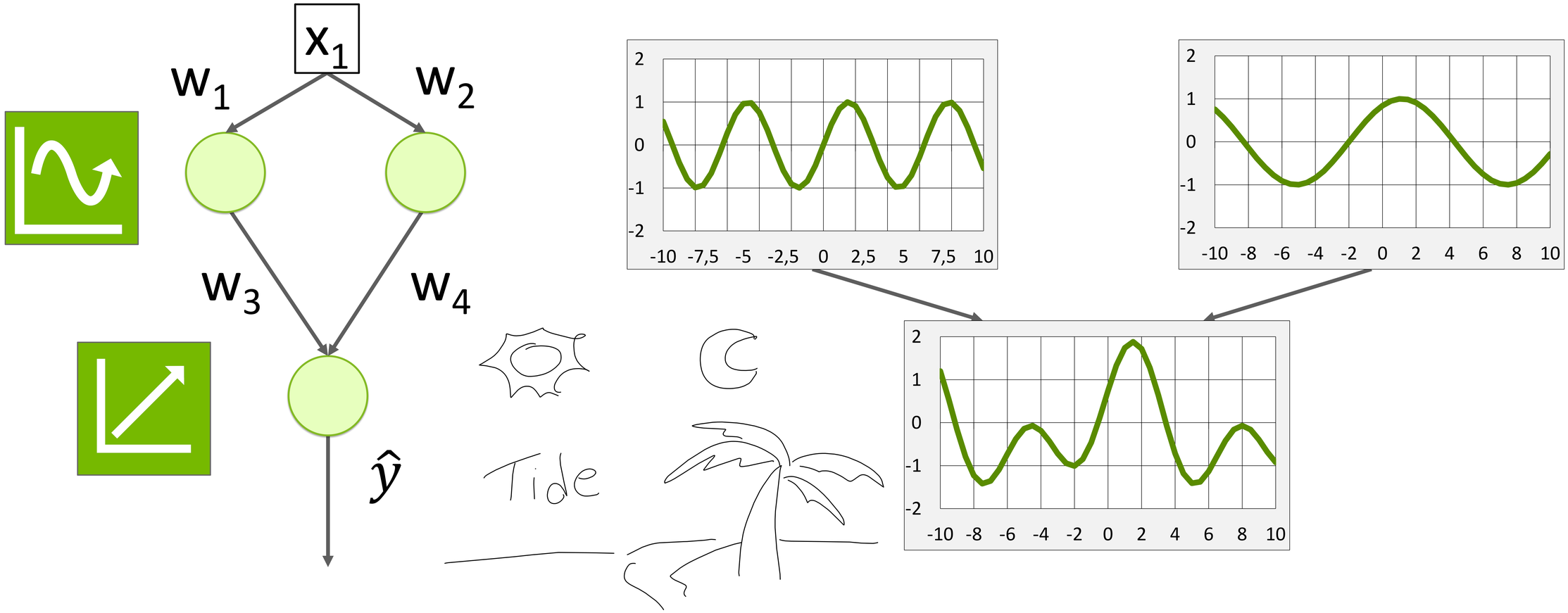
ReLU



Sigmoid



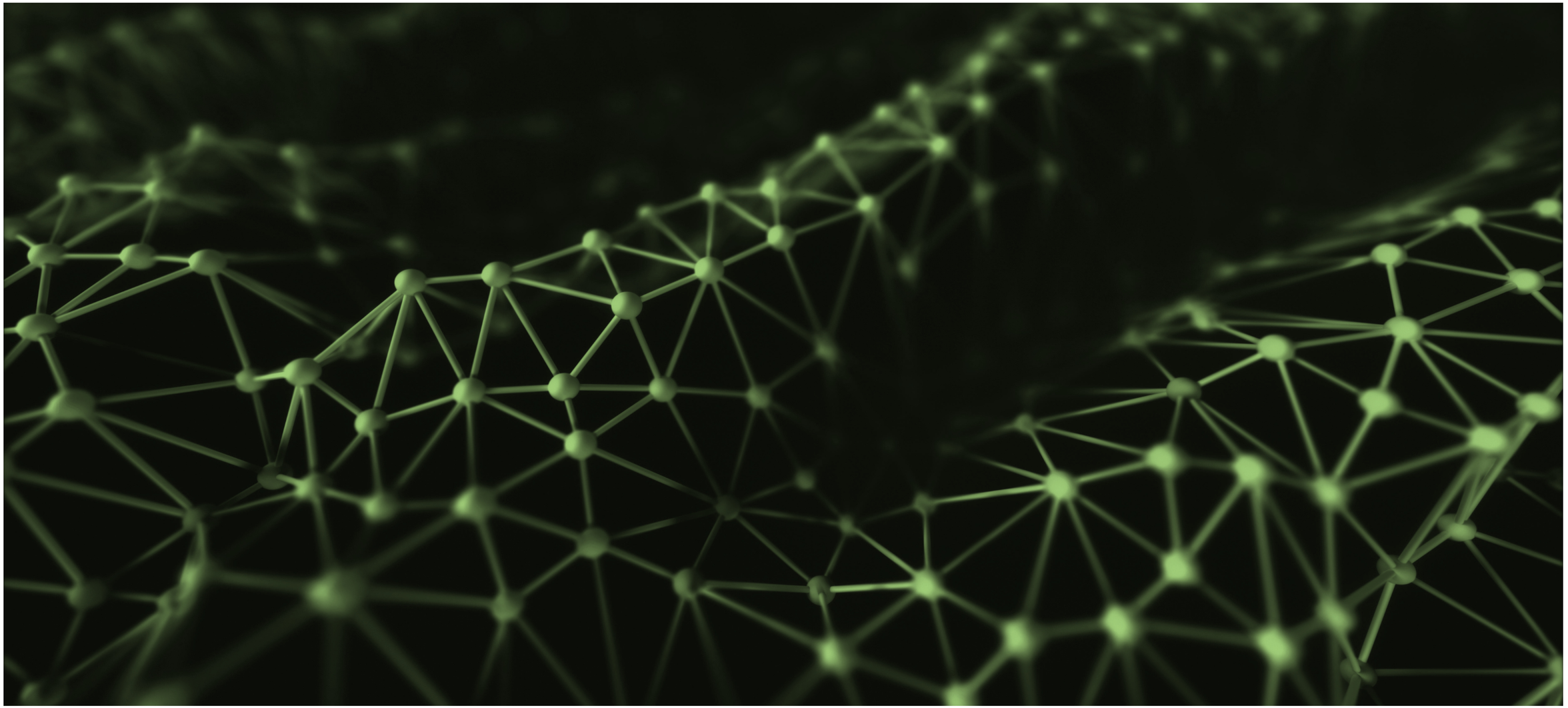
Activation Functions



Overfitting

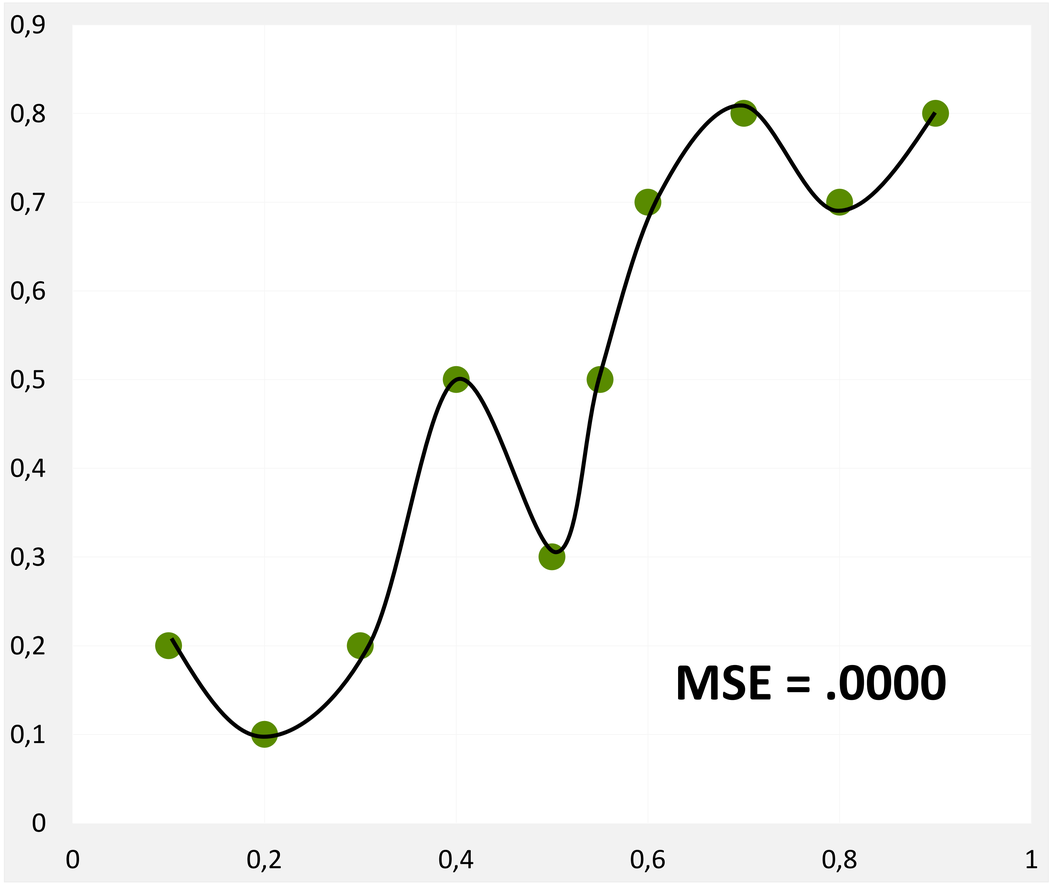
Overfitting

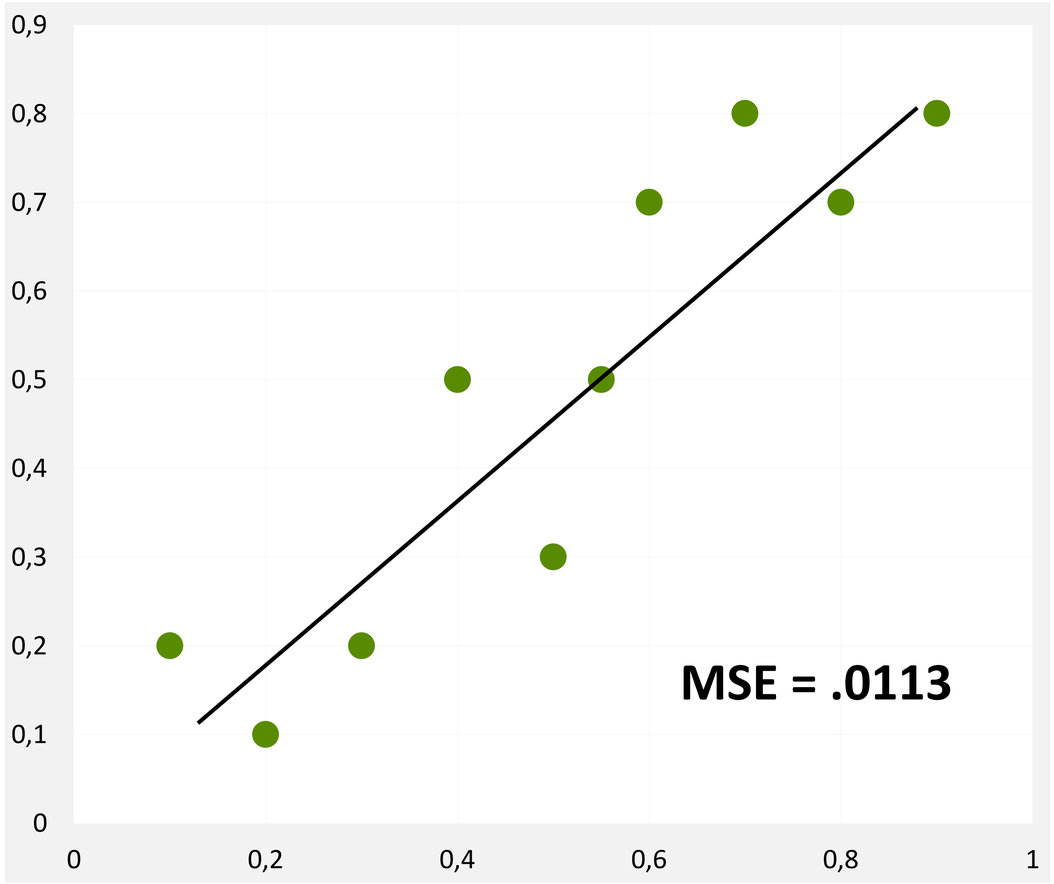
Why not have a super large neural network?



Overfitting

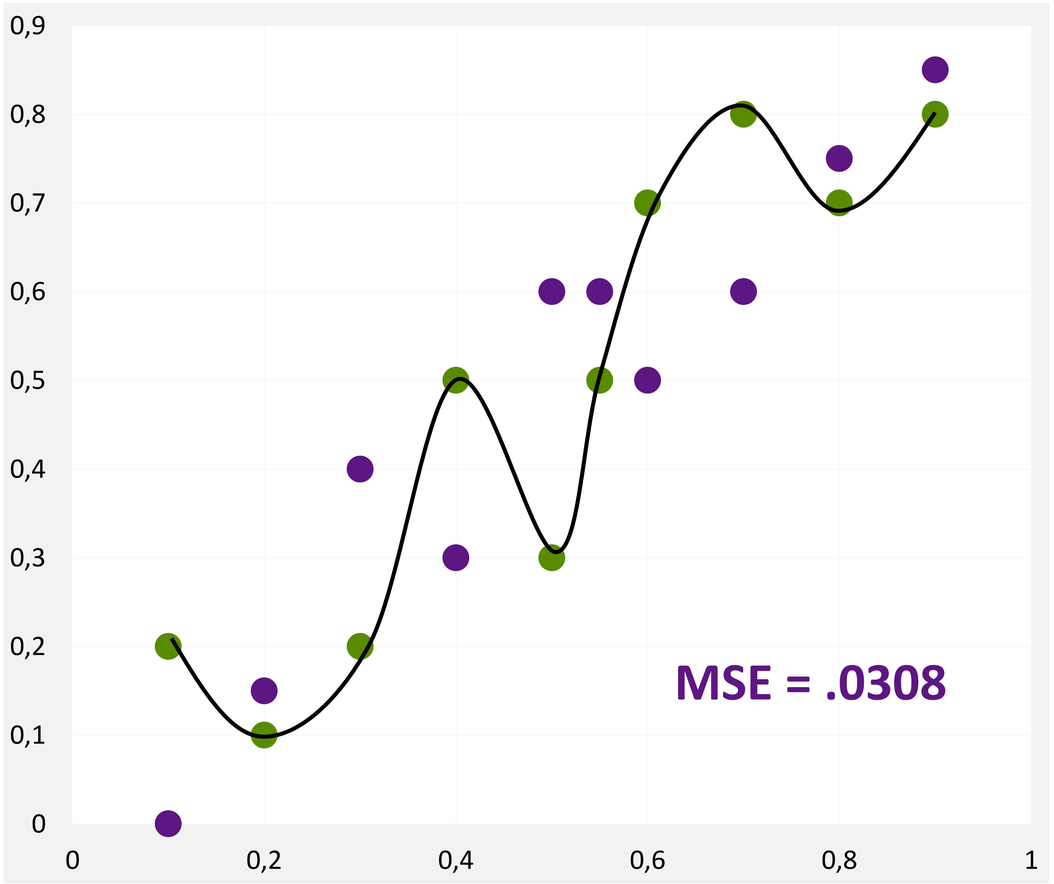
Which Trendline is Better?

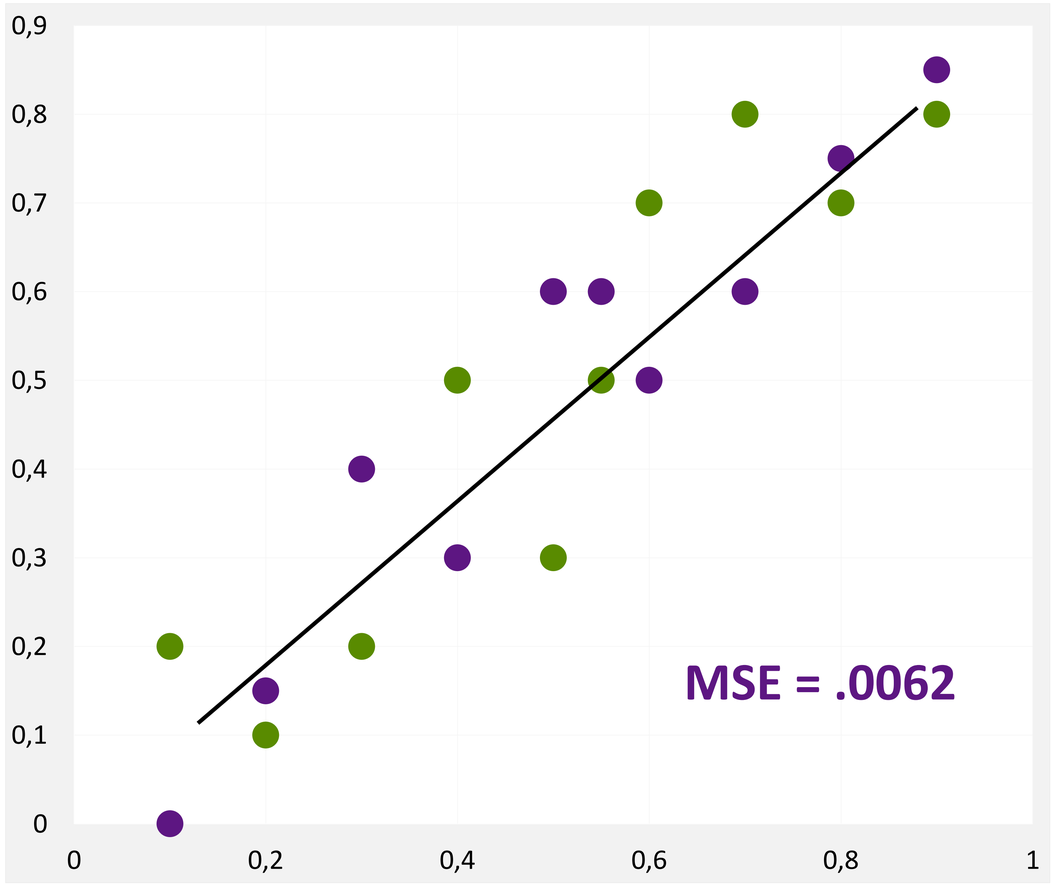




Overfitting

Which Trendline is Better?





Training vs Validation Data

Avoid memorization

Training data

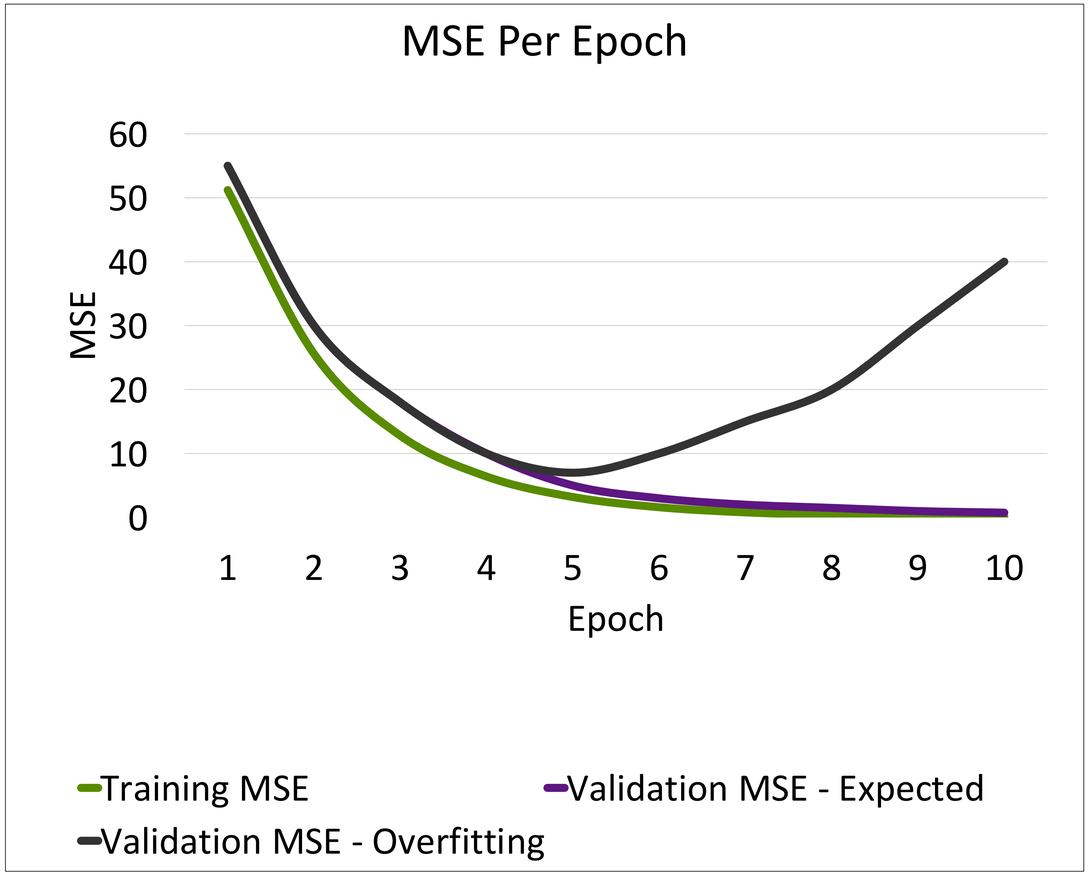
- Core dataset for the model to learn on

Validation data

- New data for model to see if it truly understands (can generalize)

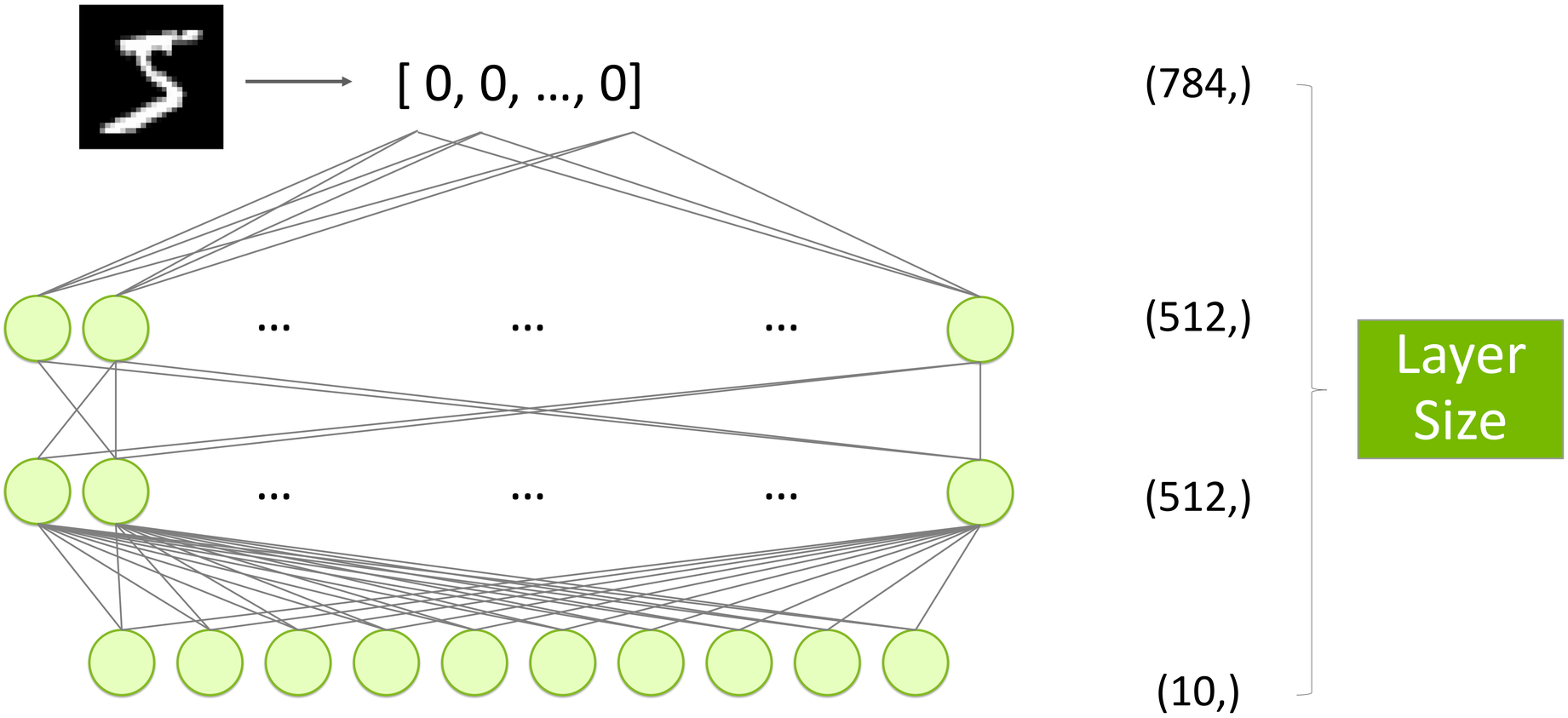
Overfitting

- When model performs well on the training data, but not the validation data (evidence of memorization)
- Ideally the accuracy and loss should be similar between both datasets



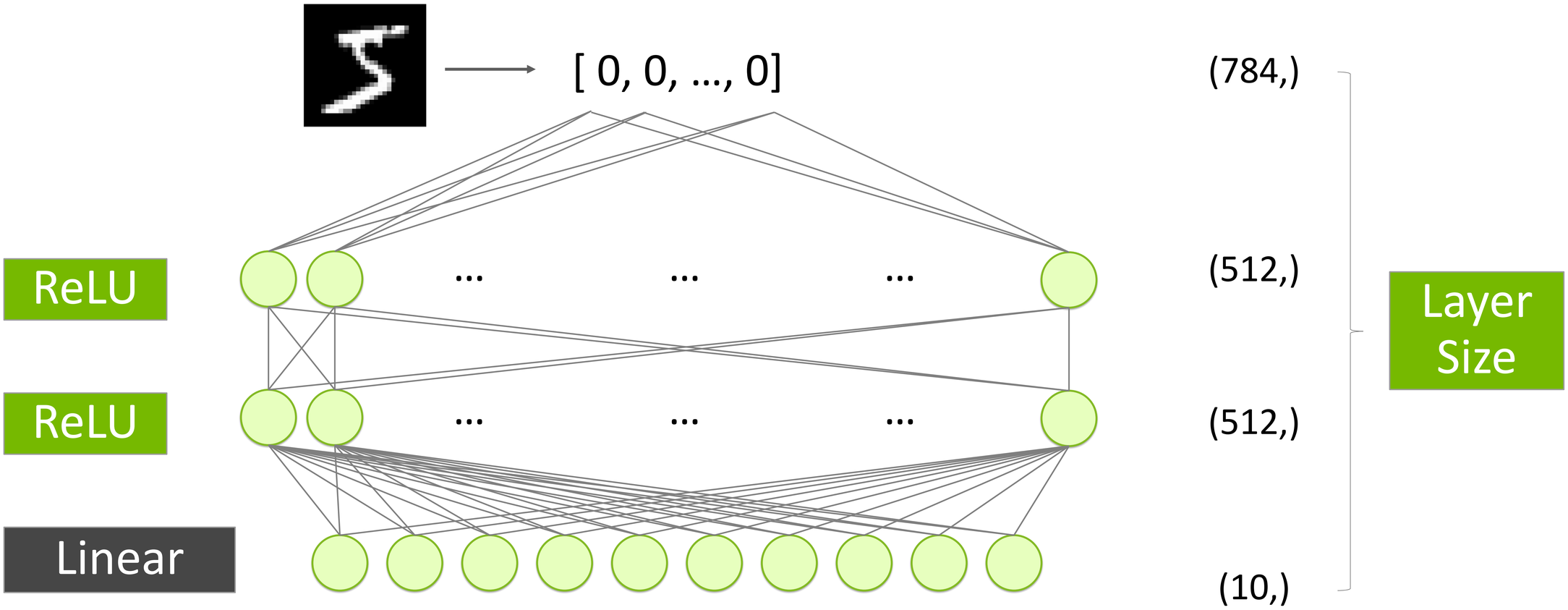
From Regression to Classification

An MNIST Model



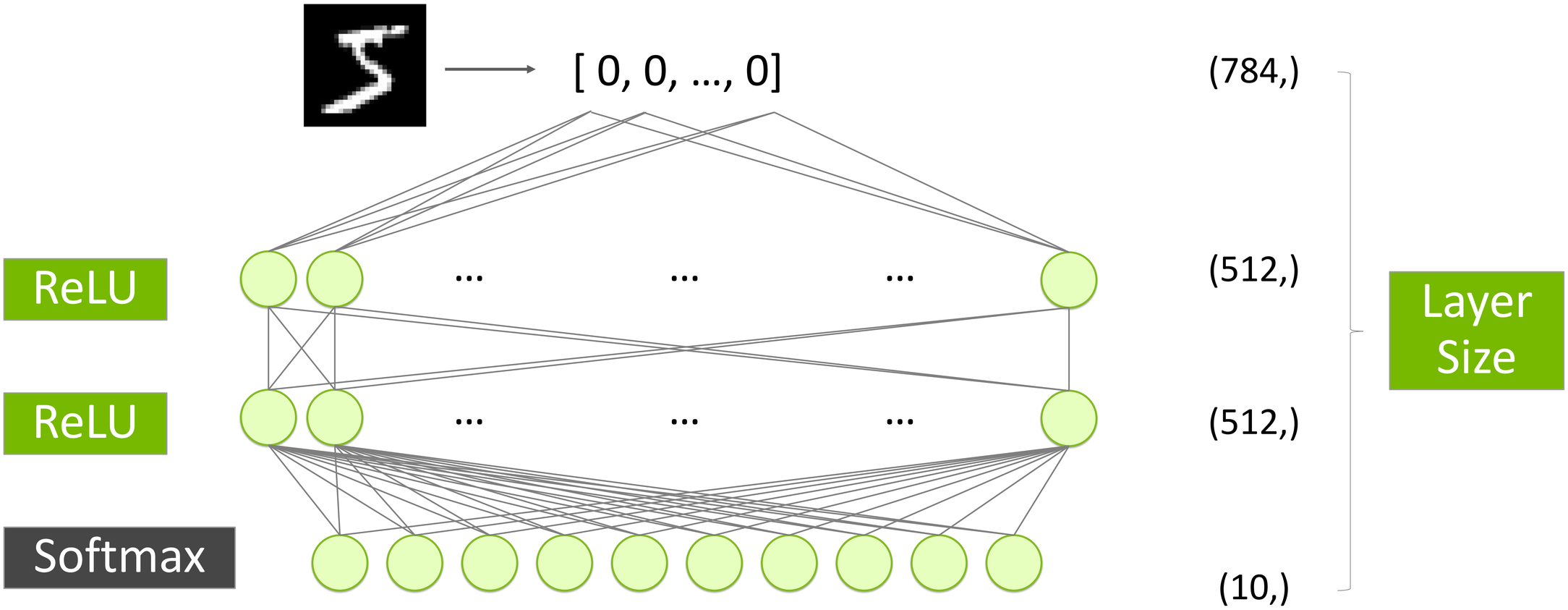
An MNIST Model

During Prediction

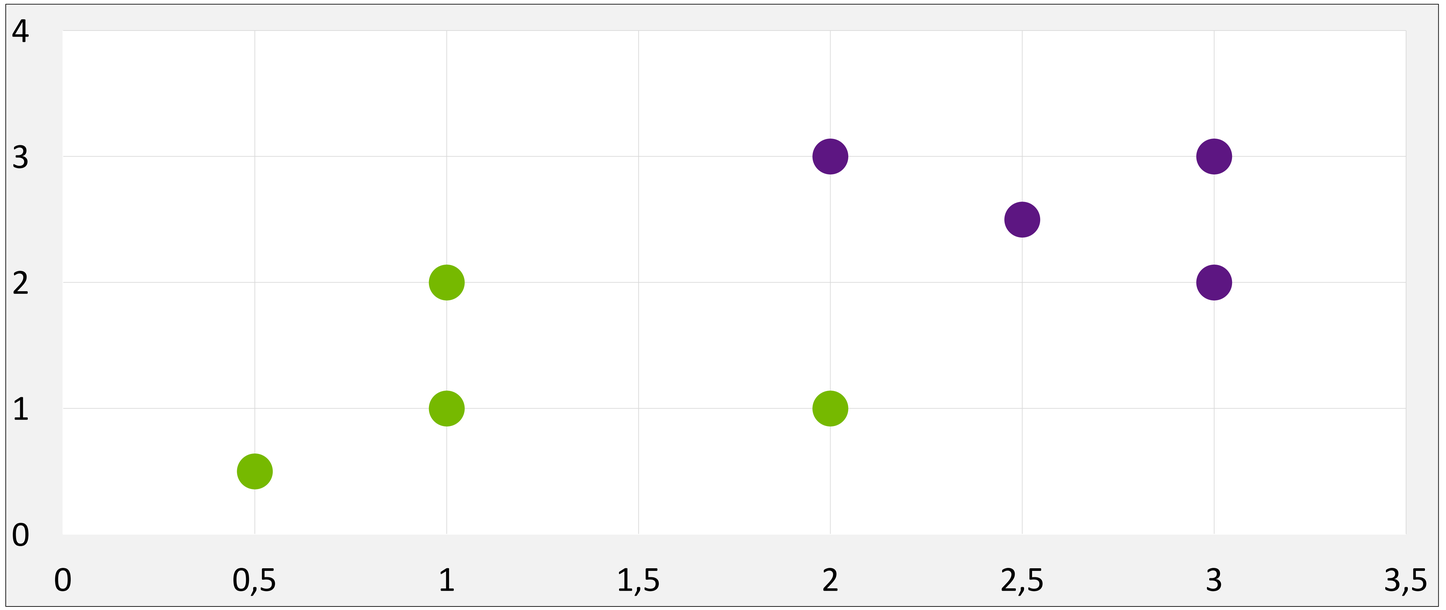


An MNIST Model

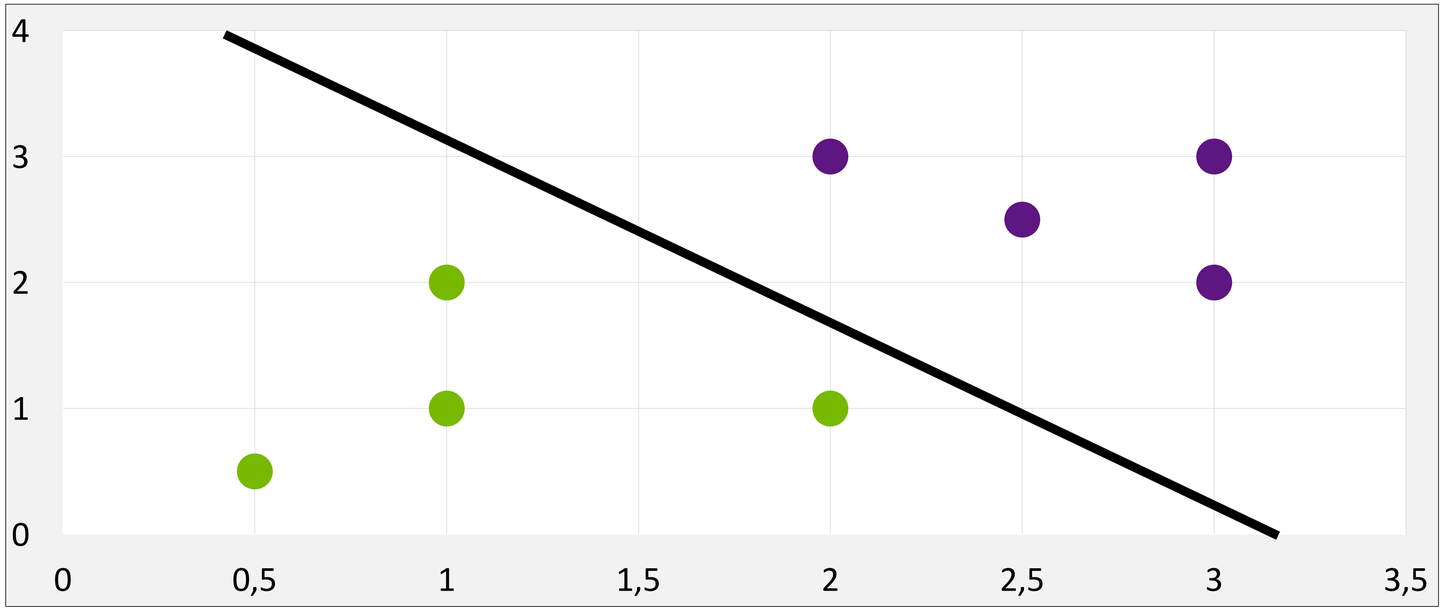
During Training



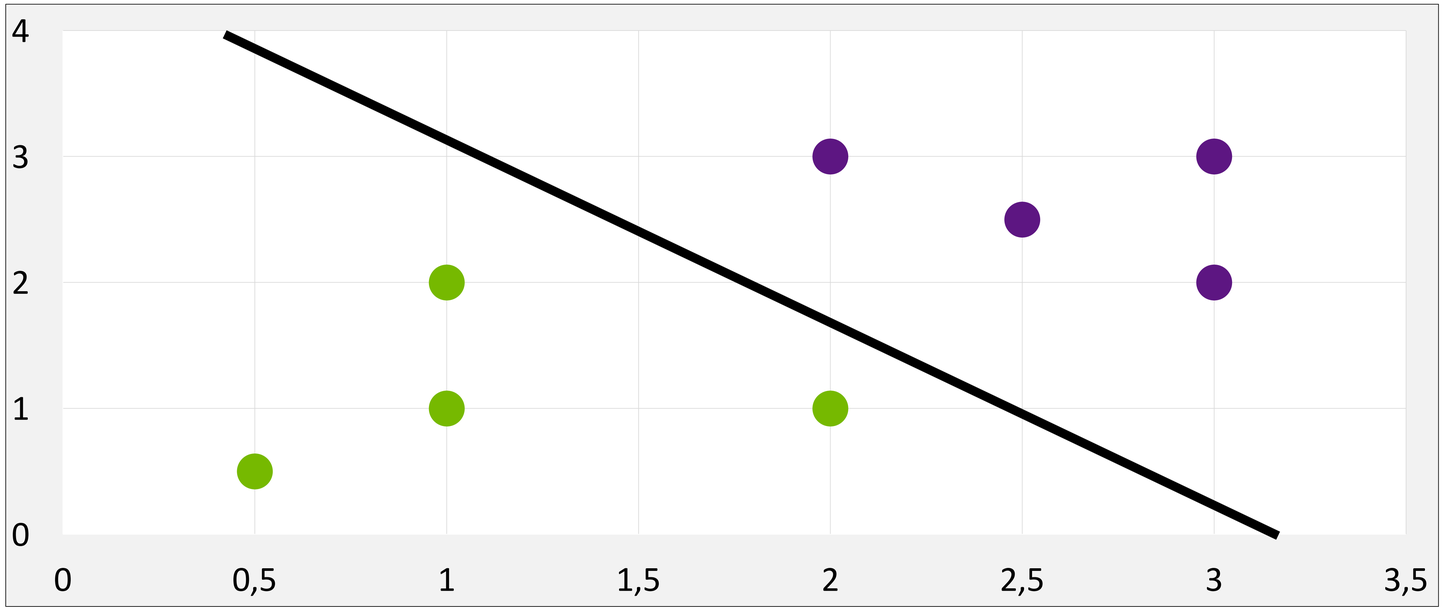
RMSE For Probabilities?

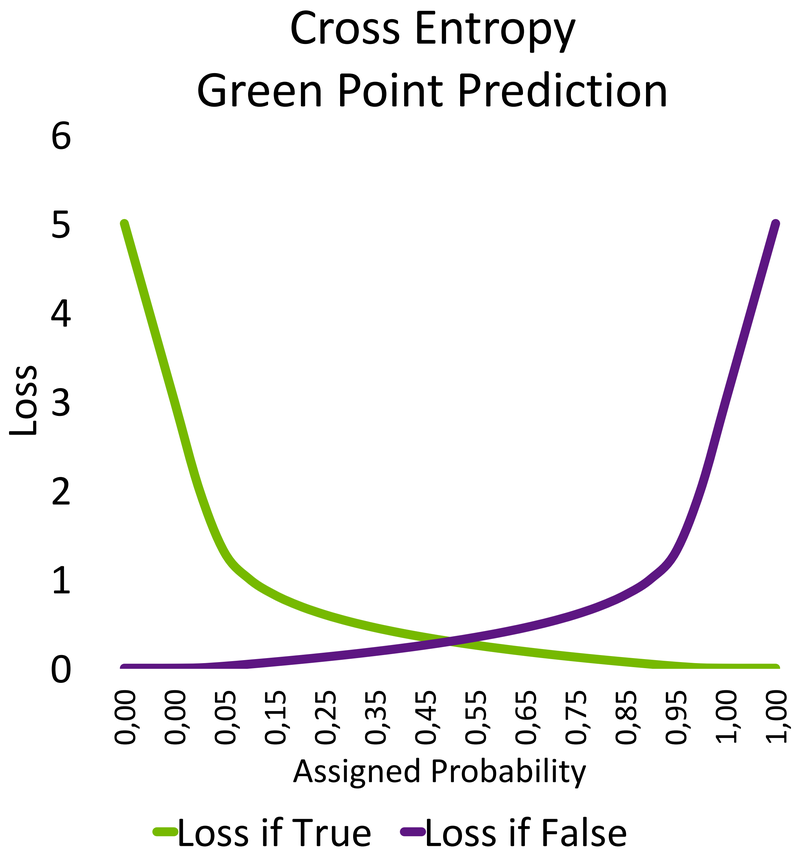


RMSE For Probabilities?

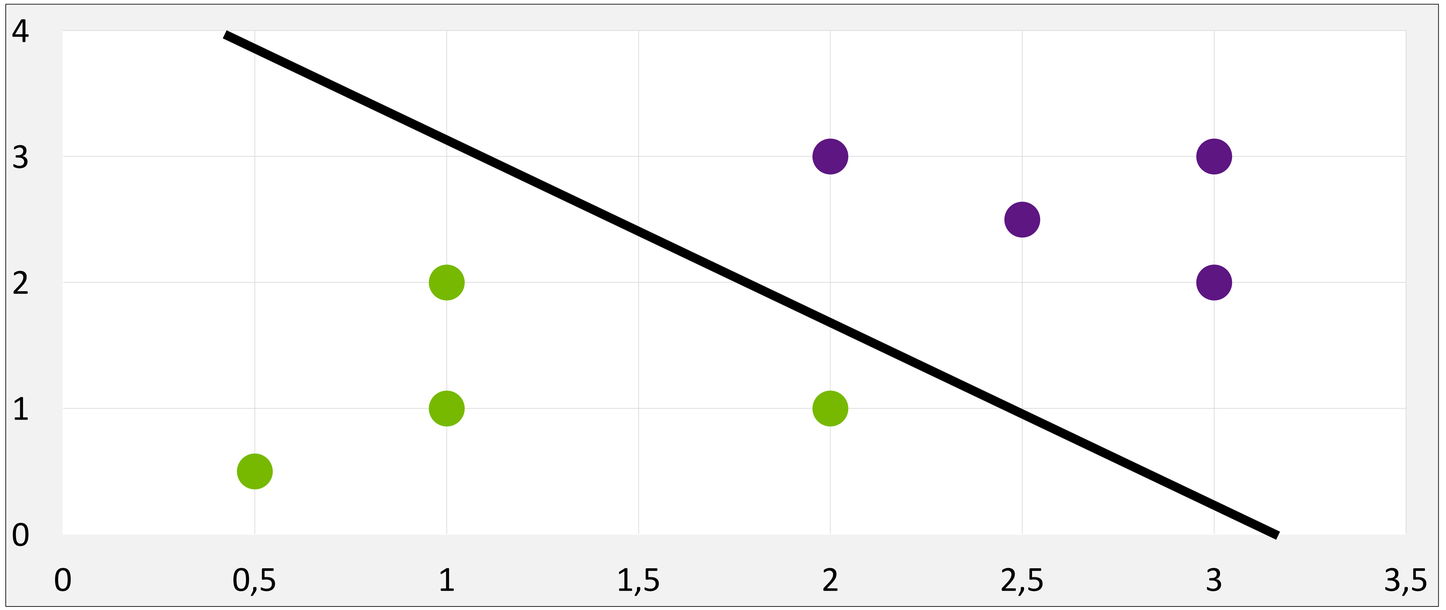


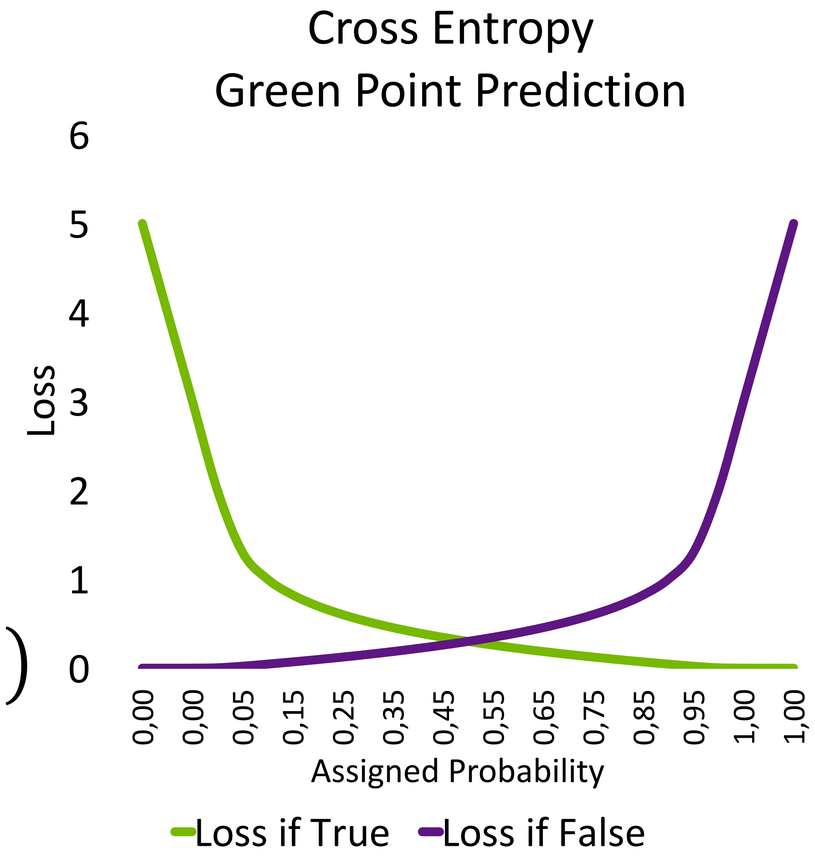
Cross Entropy





Cross Entropy



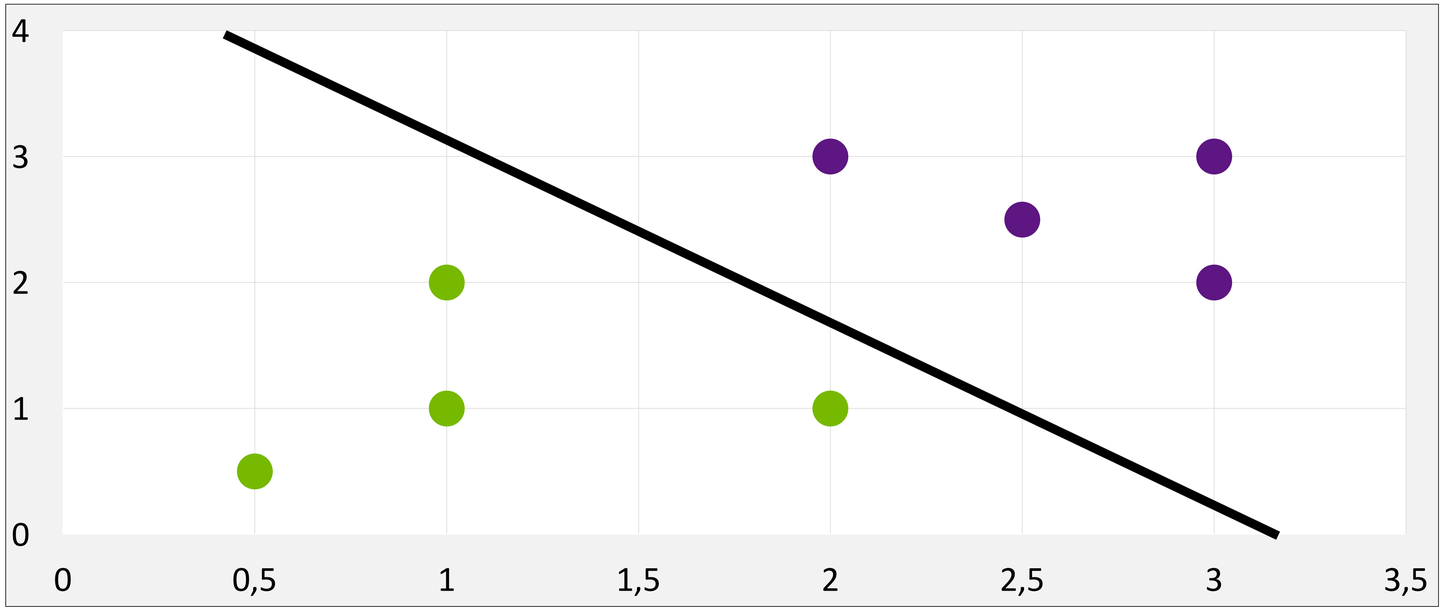


$$Loss = -\Big((t(x) \cdot \log(p(x)) + \big(1 - t(x)\big) \cdot \log(1 - p(x))\Big)$$

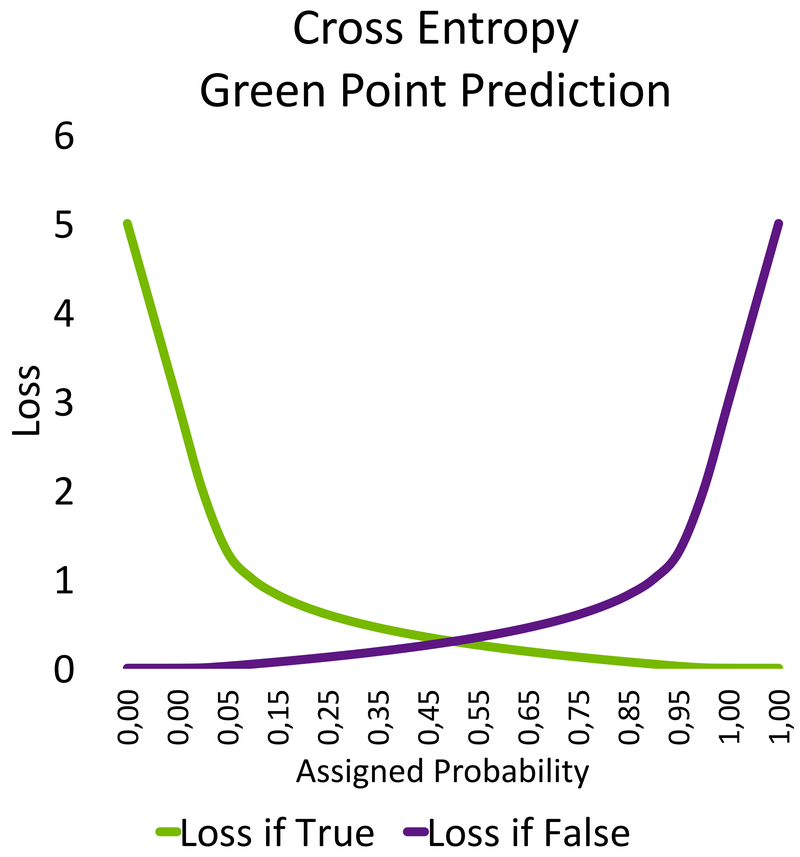
$$t(x) = target\ (0\ if\ False, 1\ if\ True)$$

$$p(x) = probability\ prediction\ of\ point\ x$$

Cross Entropy



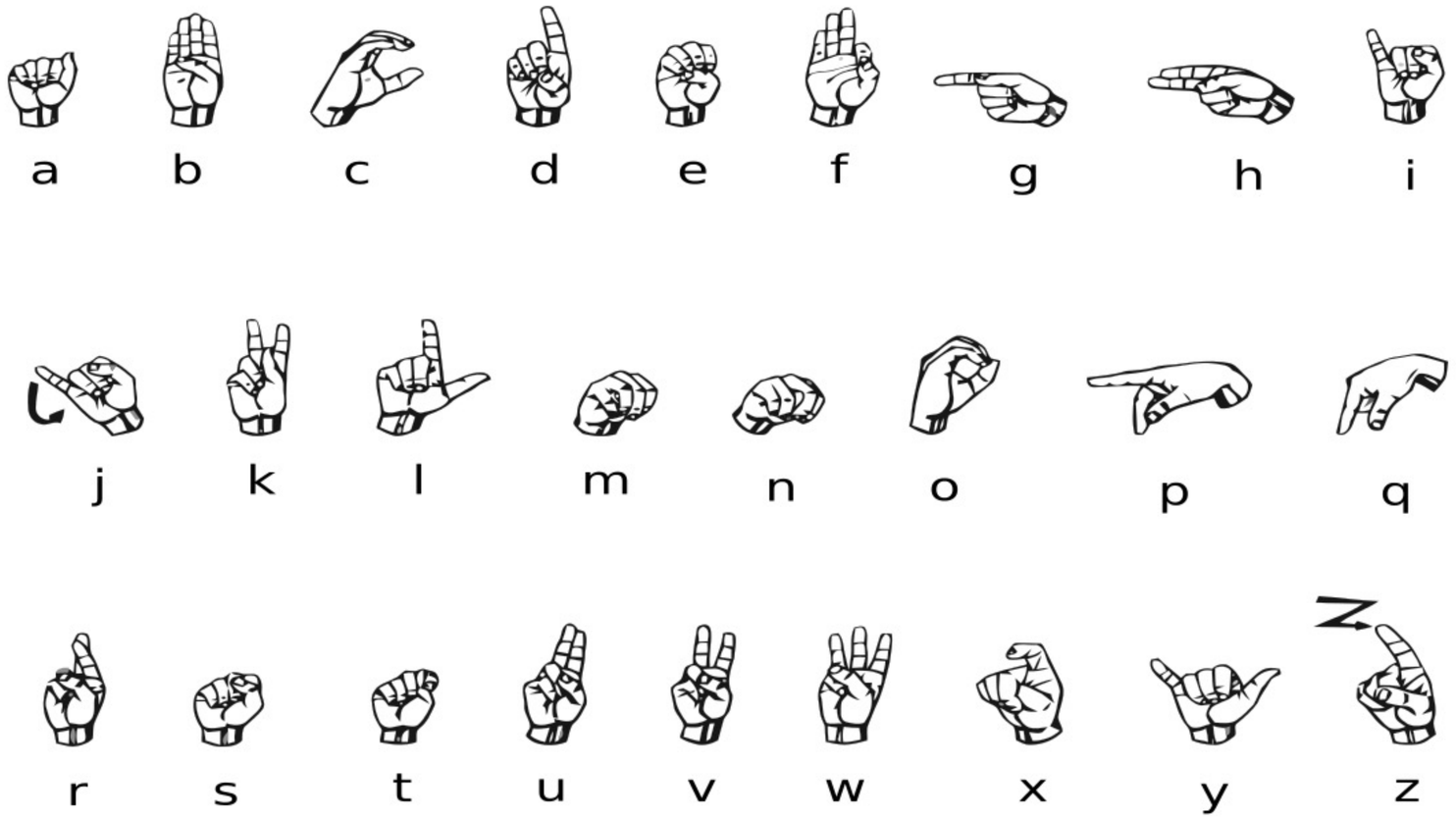

```python
def cross_entropy(y_hat, y_actual):
    """Infinite error for misplaced confidence."""
    loss = log(y_hat) if y_actual else log(1-y_hat)
    return -1*loss
```



Bringing it Together

The Next Exercise

The American Sign Language Alphabet



Let’s go!

Appendix: Gradient Descent

Helping the Computer Cheat Calculus

Learning from Error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2 = \frac{1}{n}\sum_{i=1}^{n}(y - (mx + b))^2$$

$$MSE = \frac{1}{2}((3 - (m(1) + b))^2 + (5 - (m(2) + b))^2)$$

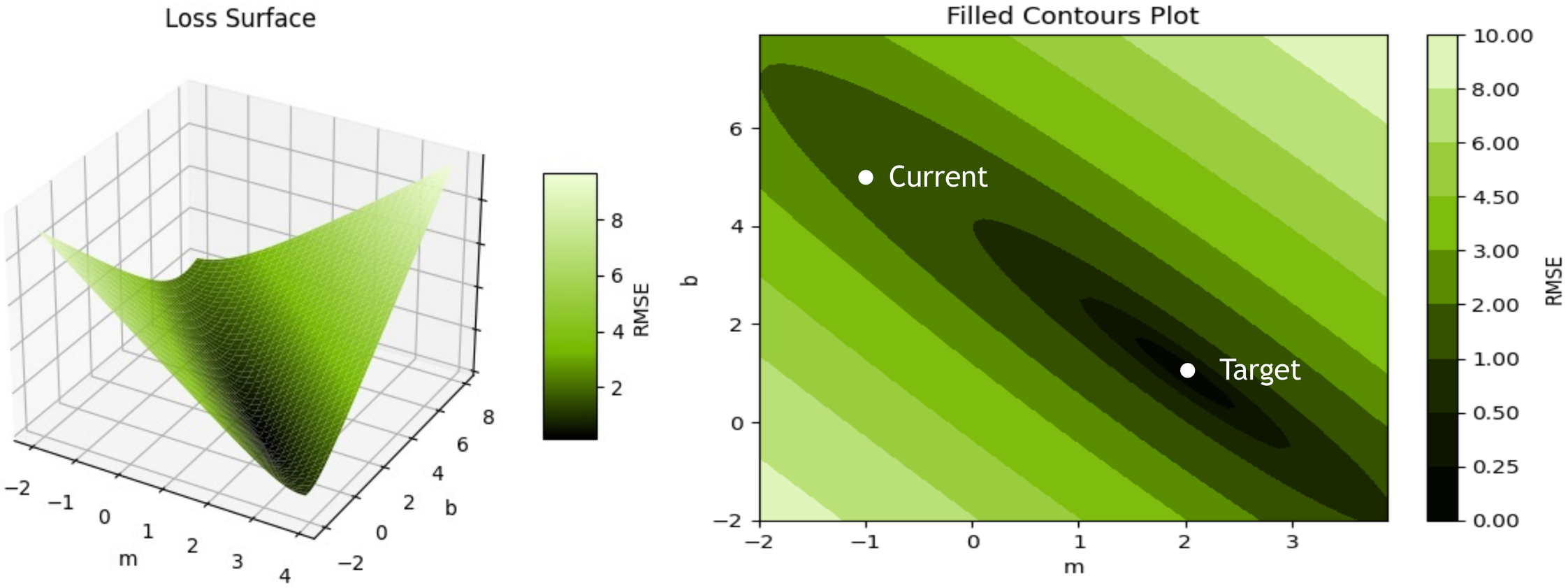
$$\frac{\partial MSE}{\partial m} = 5m + 3b - 13 \qquad \frac{\partial MSE}{\partial b} = 3m + 2b - 8$$

$$\frac{\partial MSE}{\partial m} = -3 \qquad \frac{\partial MSE}{\partial b} = -1$$

$m = -1$

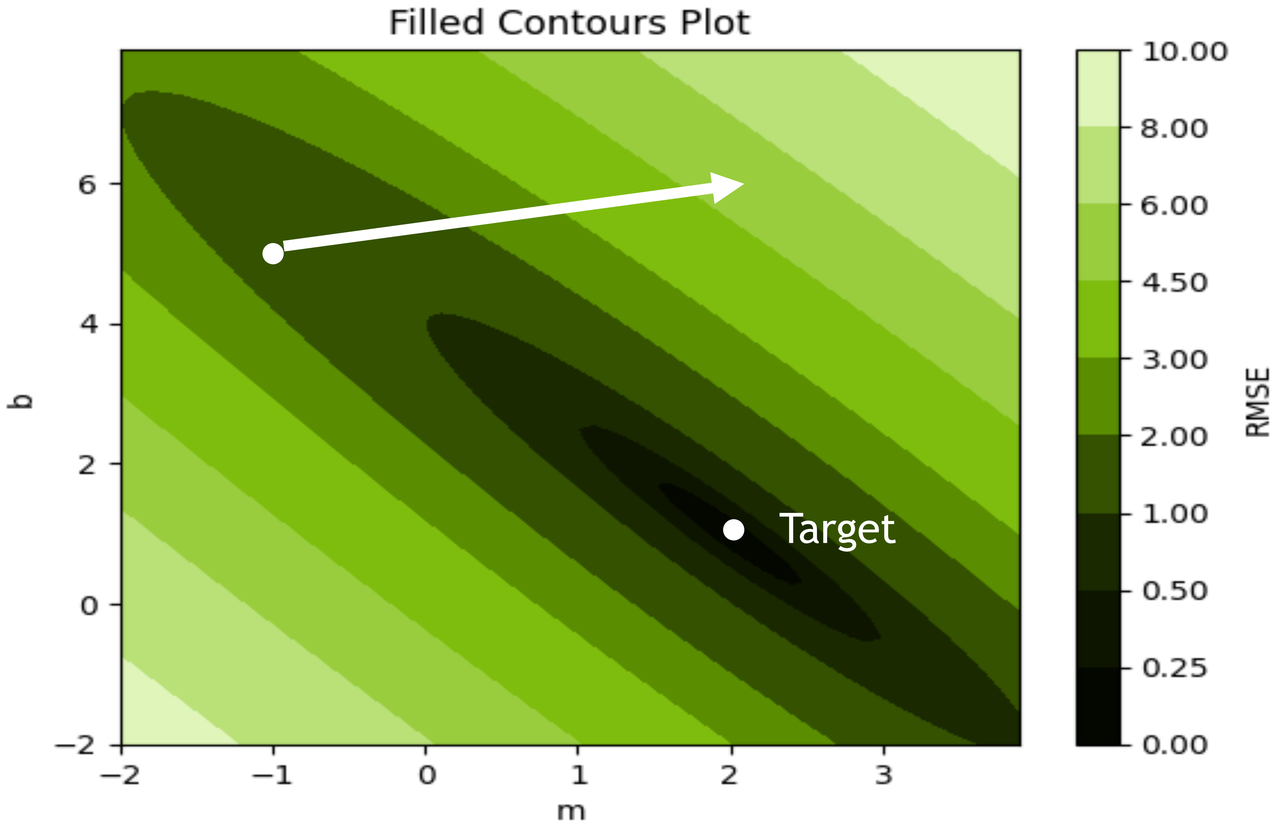
b = 5

The Loss Curve



The Loss Curve

$$\frac{\partial MSE}{\partial m} = -3 \qquad \frac{\partial MSE}{\partial b} = -1$$

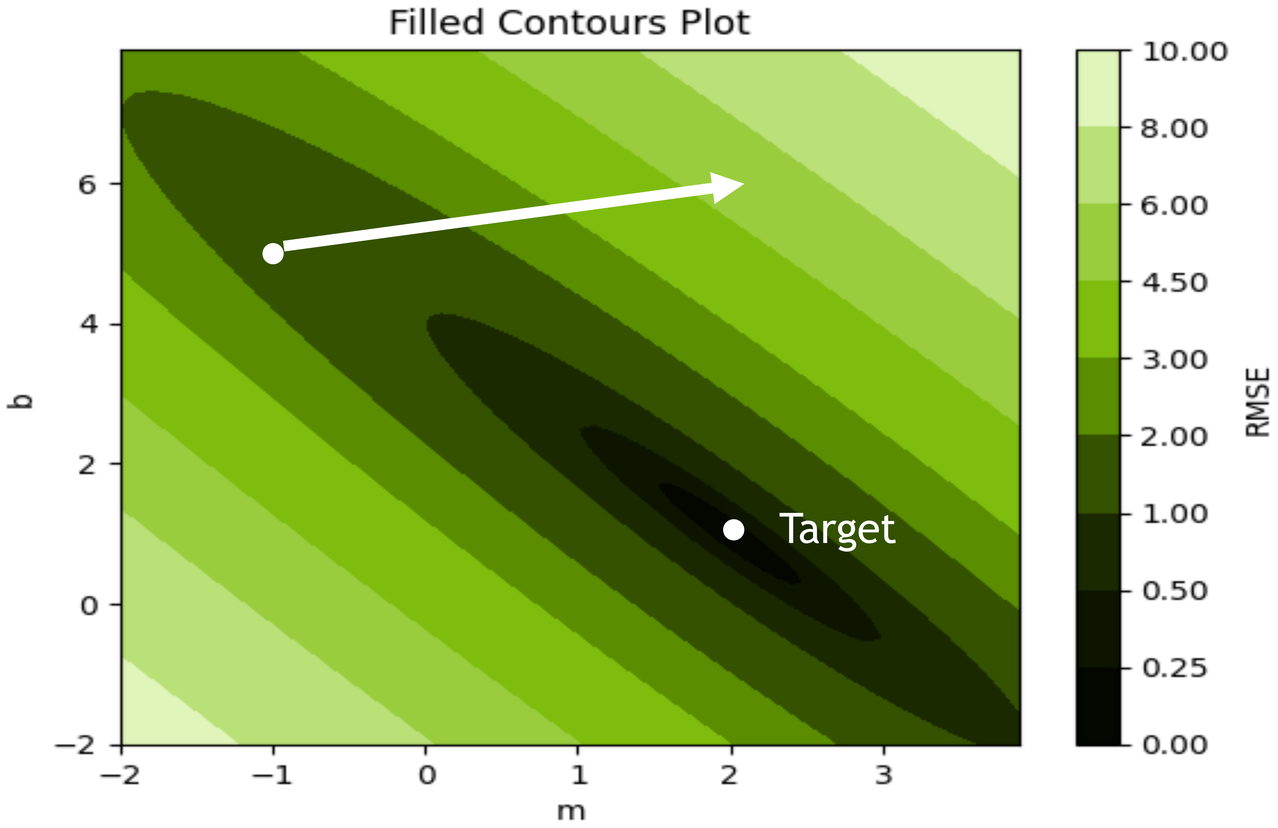


The Loss Curve

$$\frac{\partial MSE}{\partial m} = -3 \qquad \frac{\partial MSE}{\partial b} = -1$$

$$\mathrm{m} := \mathrm{m} - \lambda \frac{\partial MSE}{\partial m}$$

$$\mathrm{b} := b - \lambda \frac{\partial MSE}{\partial b}$$



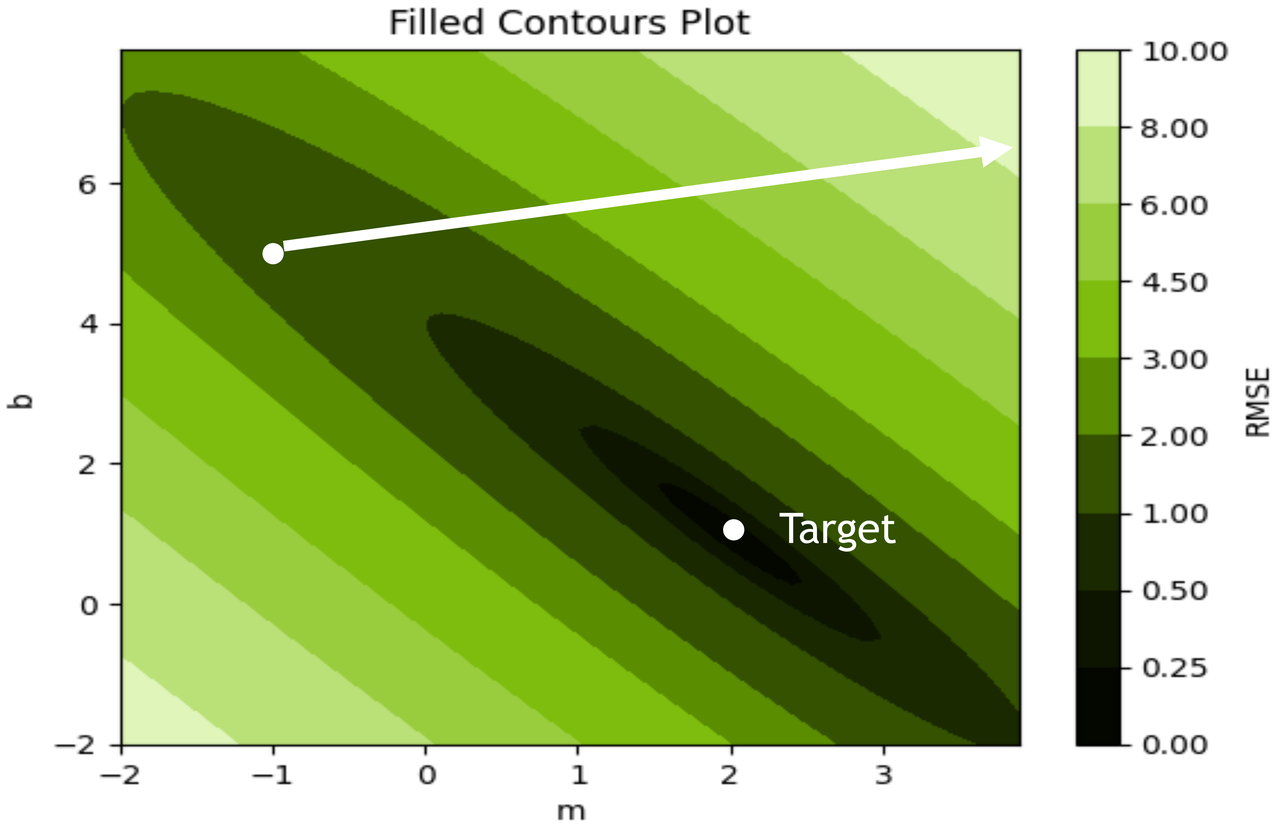
The Loss Curve

$$\frac{\partial MSE}{\partial m} = -3 \qquad \frac{\partial MSE}{\partial b} = -1$$

$$\mathrm{m} := \mathrm{m} - \lambda \frac{\partial MSE}{\partial m}$$

$$\lambda = 2$$

$$\mathrm{b} := b - \lambda \frac{\partial MSE}{\partial b}$$



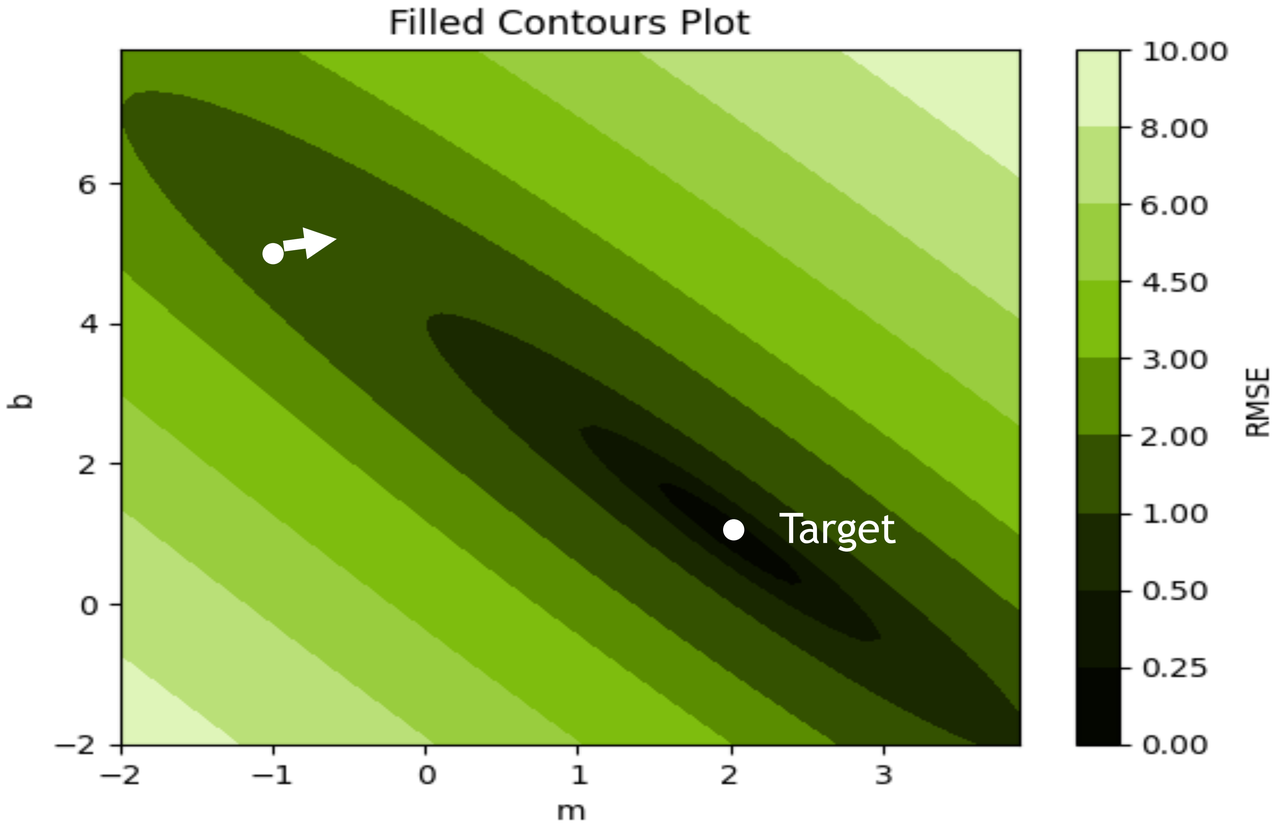
The Loss Curve

$$\frac{\partial MSE}{\partial m} = -3 \qquad \frac{\partial MSE}{\partial b} = -1$$

$$\mathrm{m} := \mathrm{m} - \lambda \frac{\partial MSE}{\partial m}$$

$$\lambda = .1$$

$$\mathrm{b} := b - \lambda \frac{\partial MSE}{\partial b}$$

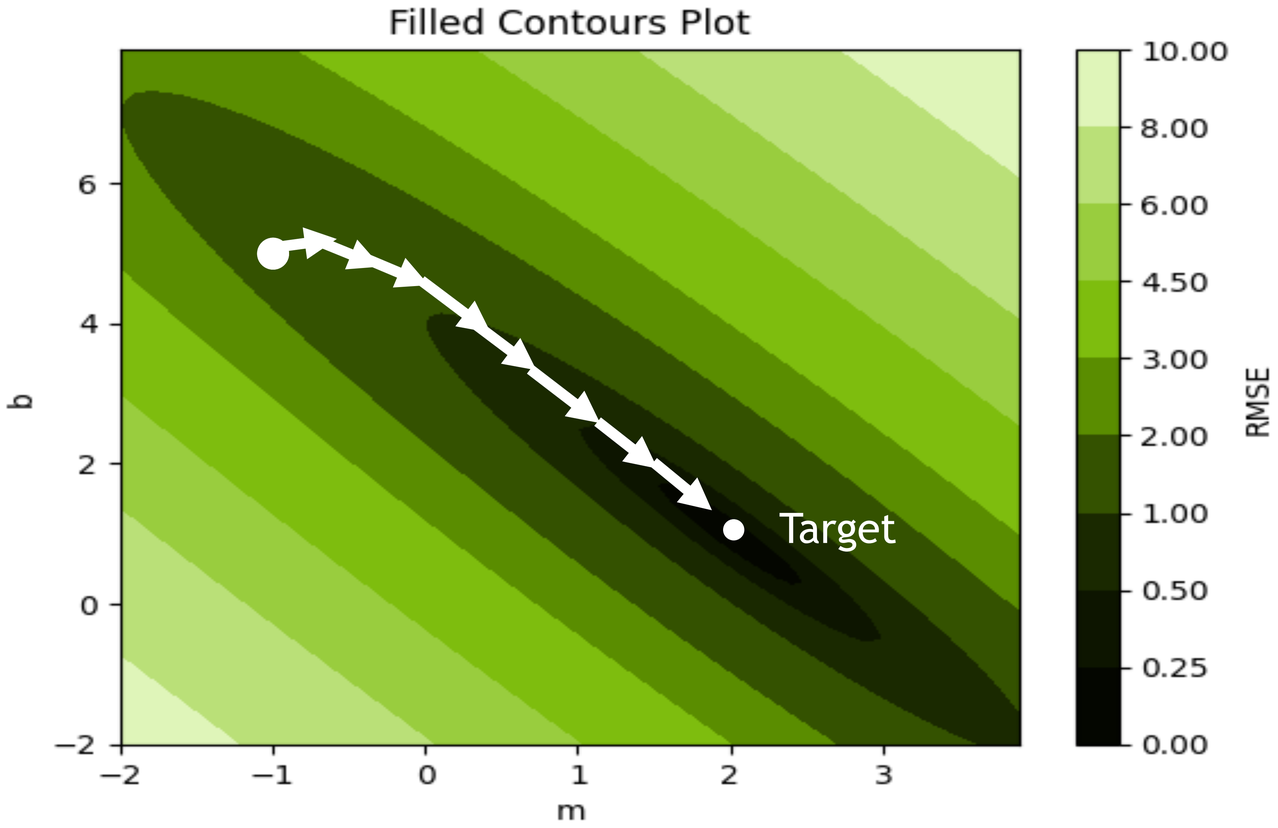


The Loss Curve

$$\lambda = .1$$

$$m := -1 + 3\,\lambda = -0.7$$

$$b := 5 + \lambda = 5.1$$



Fundamentals of Deep Learning

Part 3: Convolutional Neural Networks

Agenda

- Part 1: An Introduction to Deep Learning
- Part 2: How a Neural Network Trains
- Part 3: Convolutional Neural Networks
- Part 4: Data Augmentation and Deployment
- Part 5: Pre-Trained Models
- Part 6: Advanced Architectures

Recap of the exercise

Trained a dense neural network model

Training accuracy was high

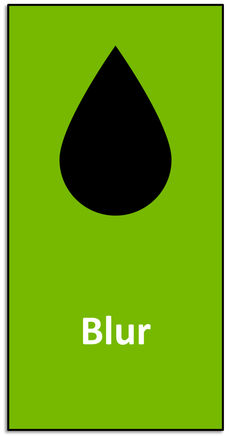
Validation accuracy was low

Evidence of overfitting

Kernels and Convolution

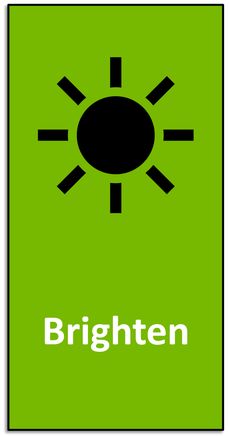
Kernels and Convolution

Original Image



Blur

Sharpen

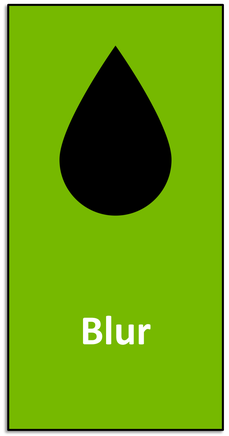


Brighten

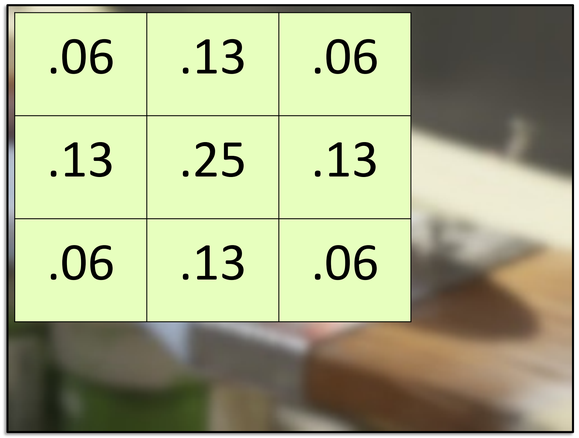
Darken

Kernels and Convolution

Original Image

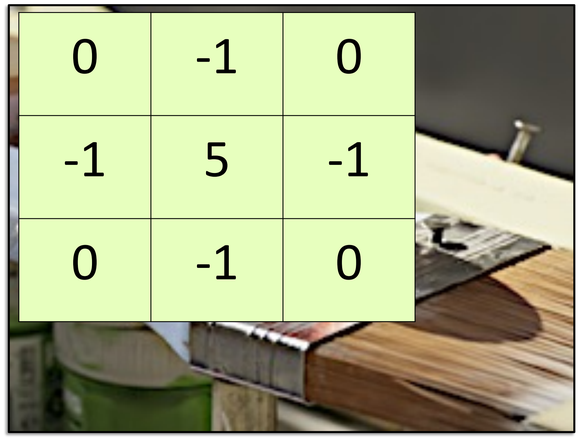


Blur

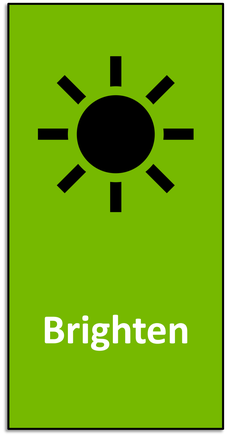


.06	.13	.06
.13	.25	.13
.06	.13	.06

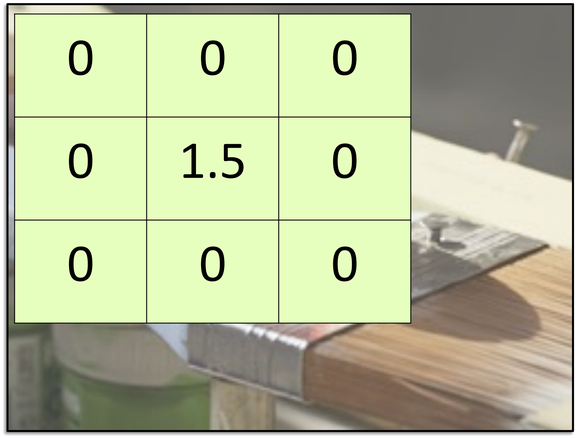
Sharpen



0	-1	0
-1	5	-1
0	-1	0

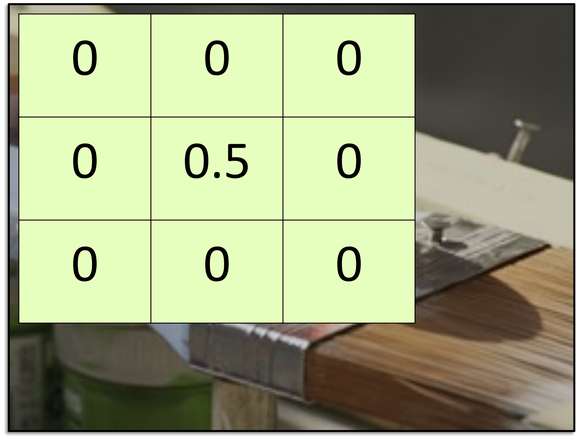


Brighten



0	0	0
0	1.5	0
0	0	0

Darken



0	0	0
0	0.5	0
0	0	0

Kernels and Convolution

Blur Kernel

.06	.13	.06
.13	.25	.13
.06	.13	.06

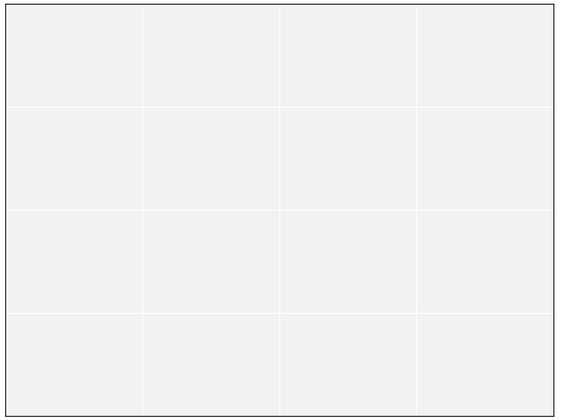
*

Original Image

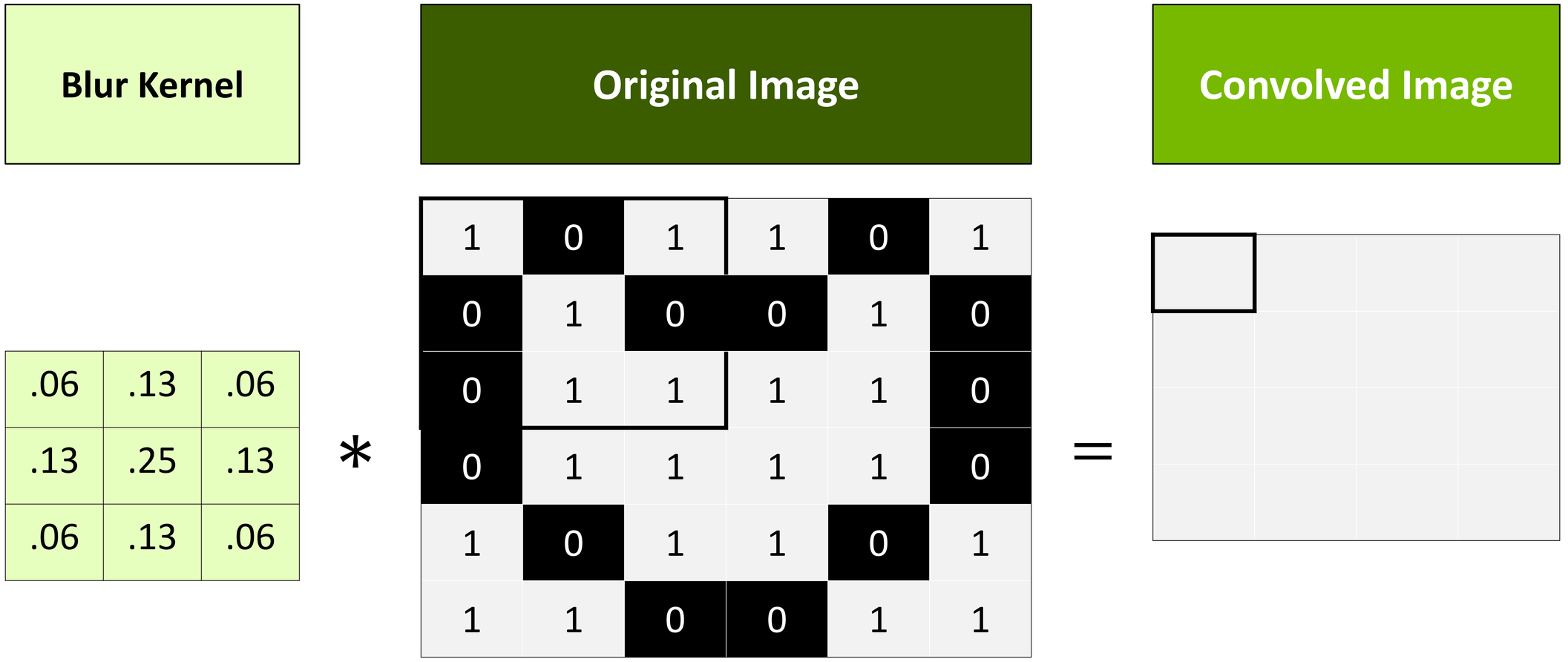
1	0	1	1	0	1
0	1	0	0	1	0
0	1	1	1	1	0
0	1	1	1	1	0
1	0	1	1	0	1
1	1	0	0	1	1

=

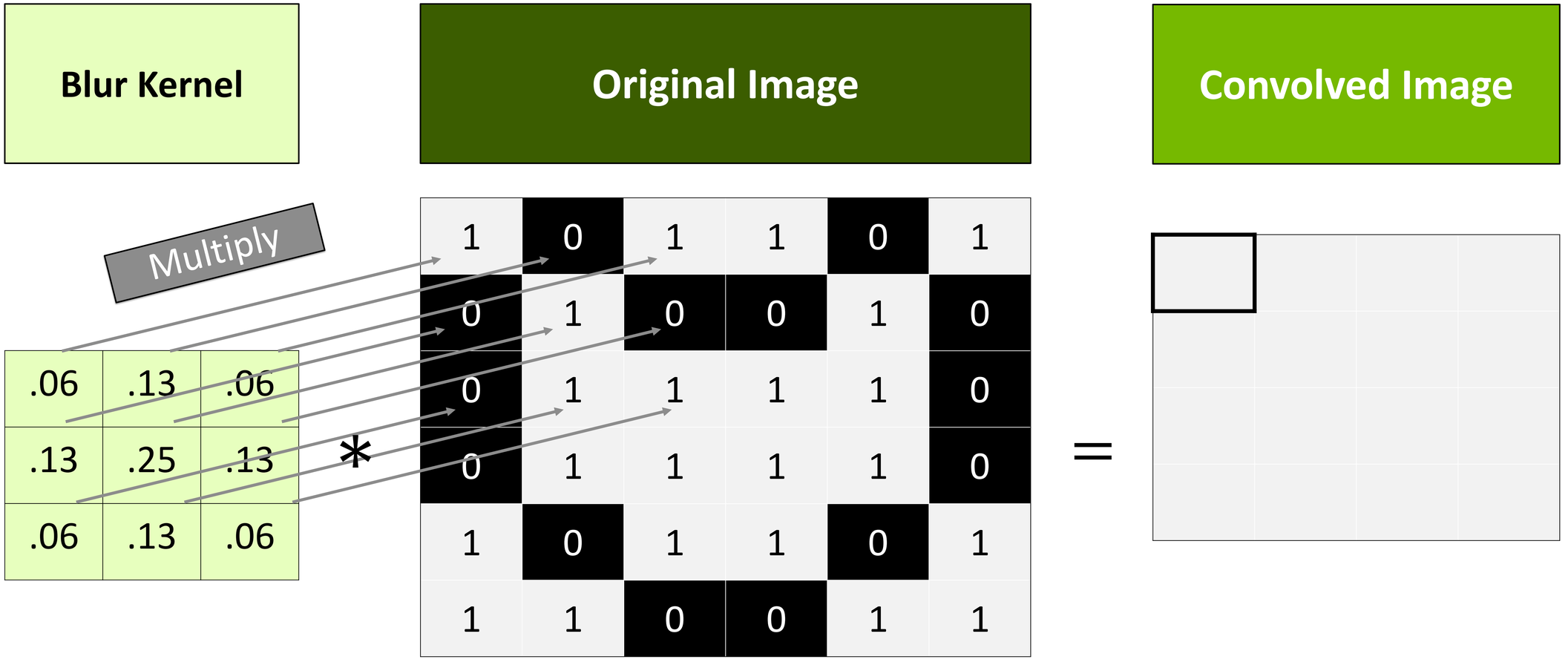
Convolved Image



Kernels and Convolution



Kernels and Convolution



Kernels and Convolution

Blur Kernel

.06	.13	.06
.13	.25	.13
.06	.13	.06

*

Original Image

.06	0	.06	1	0	1
0	.25	0	Total		0
0	.13	.06	1	1	0
0	1	1	1	1	0
1	0	1	1	0	1
1	1	0	0	1	1

=

Convolved Image

.56			

Kernels and Convolution

Blur Kernel

.06	.13	.06
.13	.25	.13
.06	.13	.06

*

Original Image

1	0	.13	.06	0	1
0	.13	0	0	1	0
0	.06	.13	.06	1	0
0	1	1	1	1	0
1	0	1	1	0	1
1	1	0	0	1	1

=

Convolved Image

.56	**.57**		

Kernels and Convolution

Blur Kernel

.06	.13	.06
.13	.25	.13
.06	.13	.06

*

Original Image

1	0	1	1	0	1
0	1	0	0	1	0
0	1	1	1	1	0
0	1	1	1	1	0
1	0	1	1	0	1
1	1	0	0	1	1

=

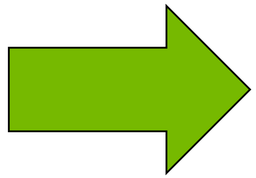
Convolved Image

.56	.57	.57	.56
.7	.82	.82	.7
.69	.95	.95	.69
.64	.69	.69	.64

Stride

Stride 1

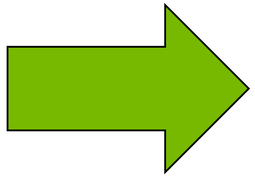
1	0	1	1	0	1
0	1	0	0	1	0
0	1	1	1	1	0



.56	.57	.57	.56

Stride 2

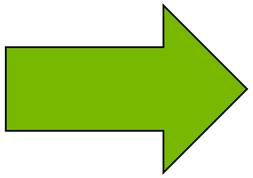
1	0	1	1	0	1
0	1	0	0	1	0
0	1	1	1	1	0



.56	.57

Stride 3

1	0	1	1	0	1
0	1	0	0	1	0
0	1	1	1	1	0



.56	.56

Padding

Original Image

1	0	1	1	0	1
0	1	0	0	1	0
0	1	1	1	1	0
0	1	1	1	1	0
1	0	1	1	0	1
1	1	0	0	1	1

Zero Padding

0	0	0	0	0	0	0	0
0	1	0	1	1	0	1	0
0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	0
0	0	0	0	0	0	0	0

Padding

Original Image

1	0	1	1	0	1
0	1	0	0	1	0
0	1	1	1	1	0
0	1	1	1	1	0
1	0	1	1	0	1
1	1	0	0	1	1

Mirror Padding

1	1	0	1	1	0	1	1
1	1	0	1	1	0	1	1
0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
1	1	0	1	1	0	1	1
1	1	1	0	0	1	1	1
1	1	1	0	0	1	1	1

Kernels and Neural Networks

Kernels and Neural Networks

Kernel

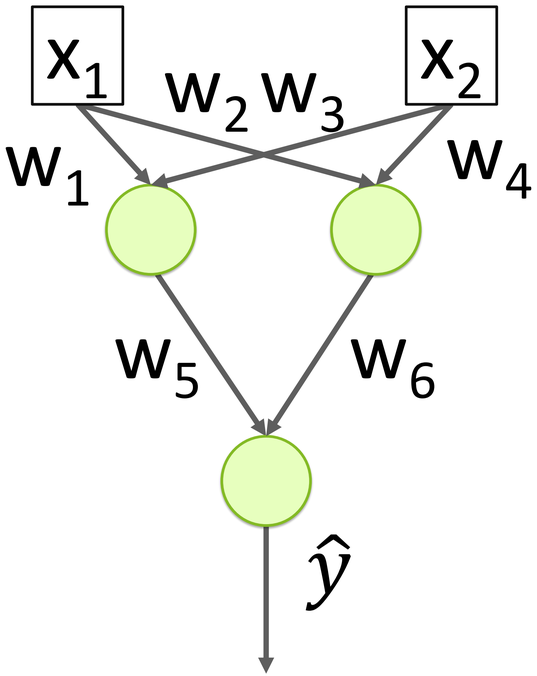
w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Kernels and Neural Networks

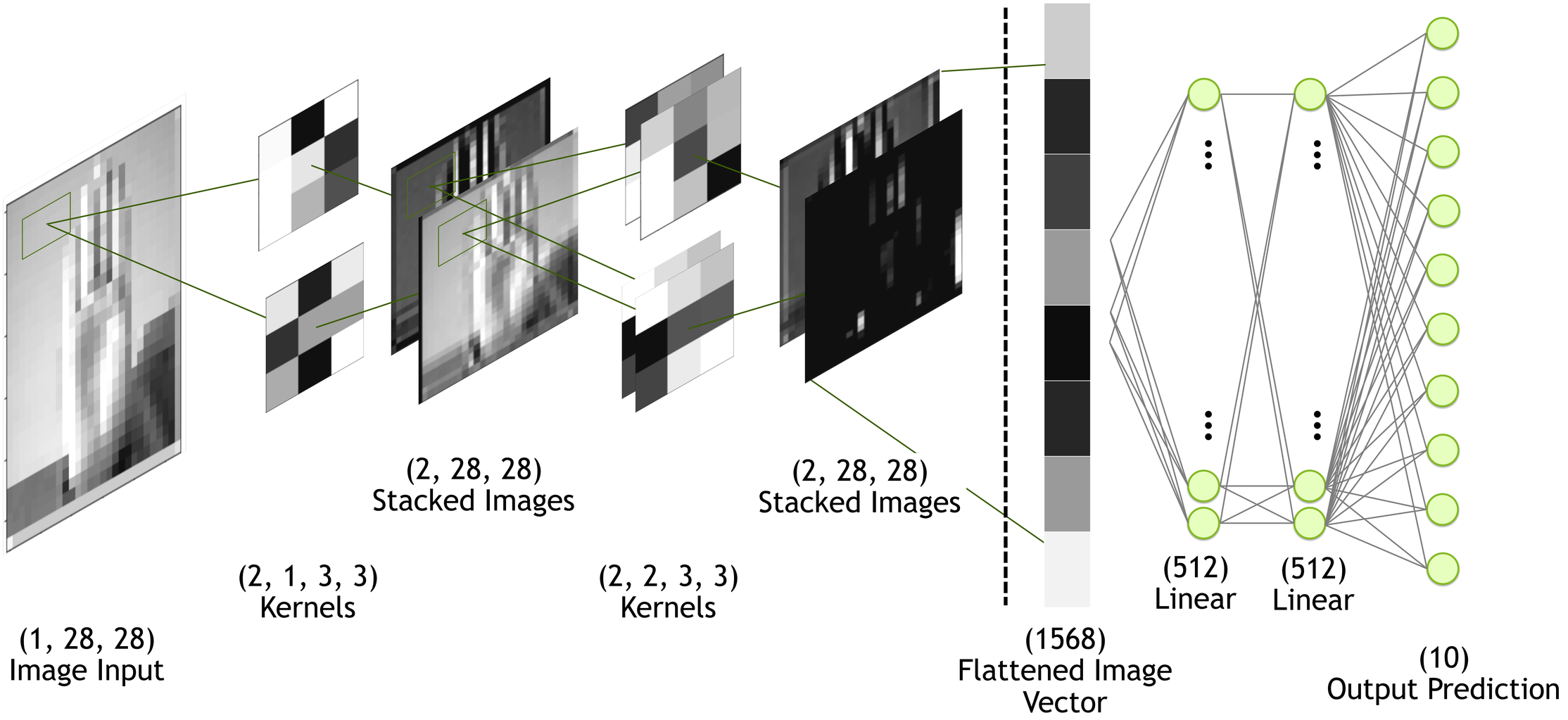
Kernel

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Neuron

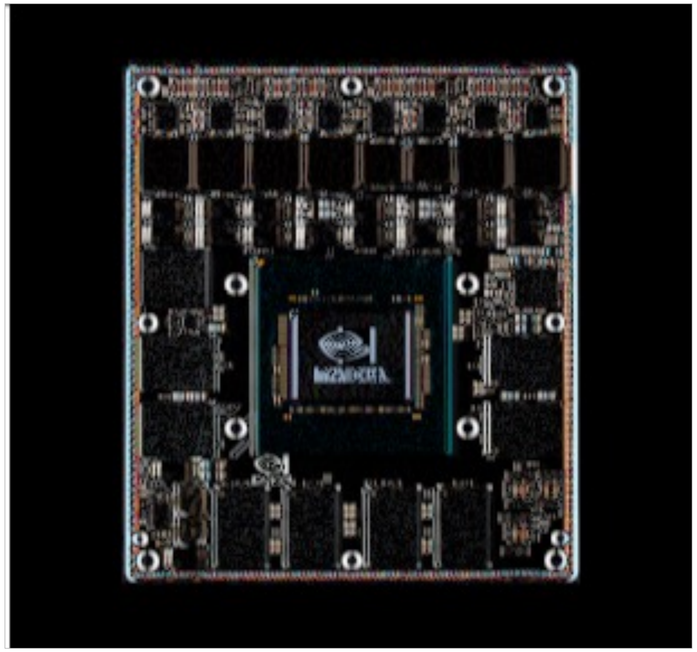


Kernels and Neural Networks



Finding Edges

Vertical Edges

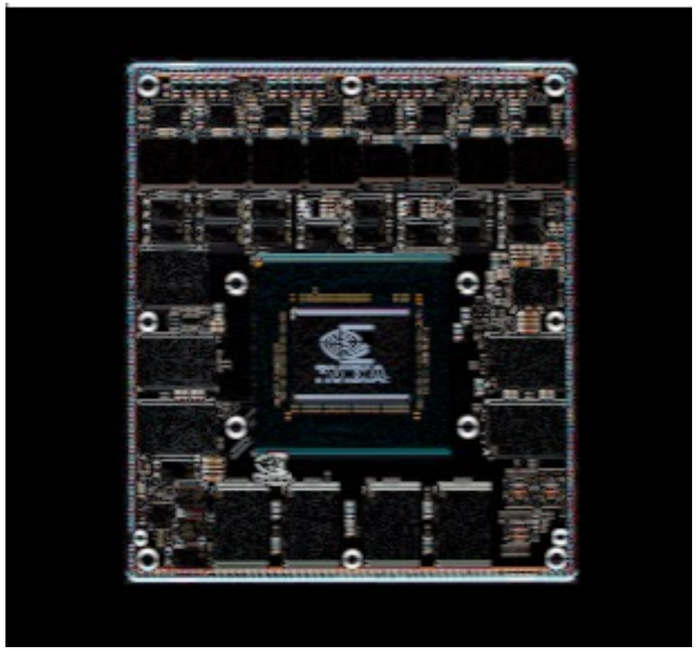


1	0	-1
2	0	-2
1	0	-1

Original Image

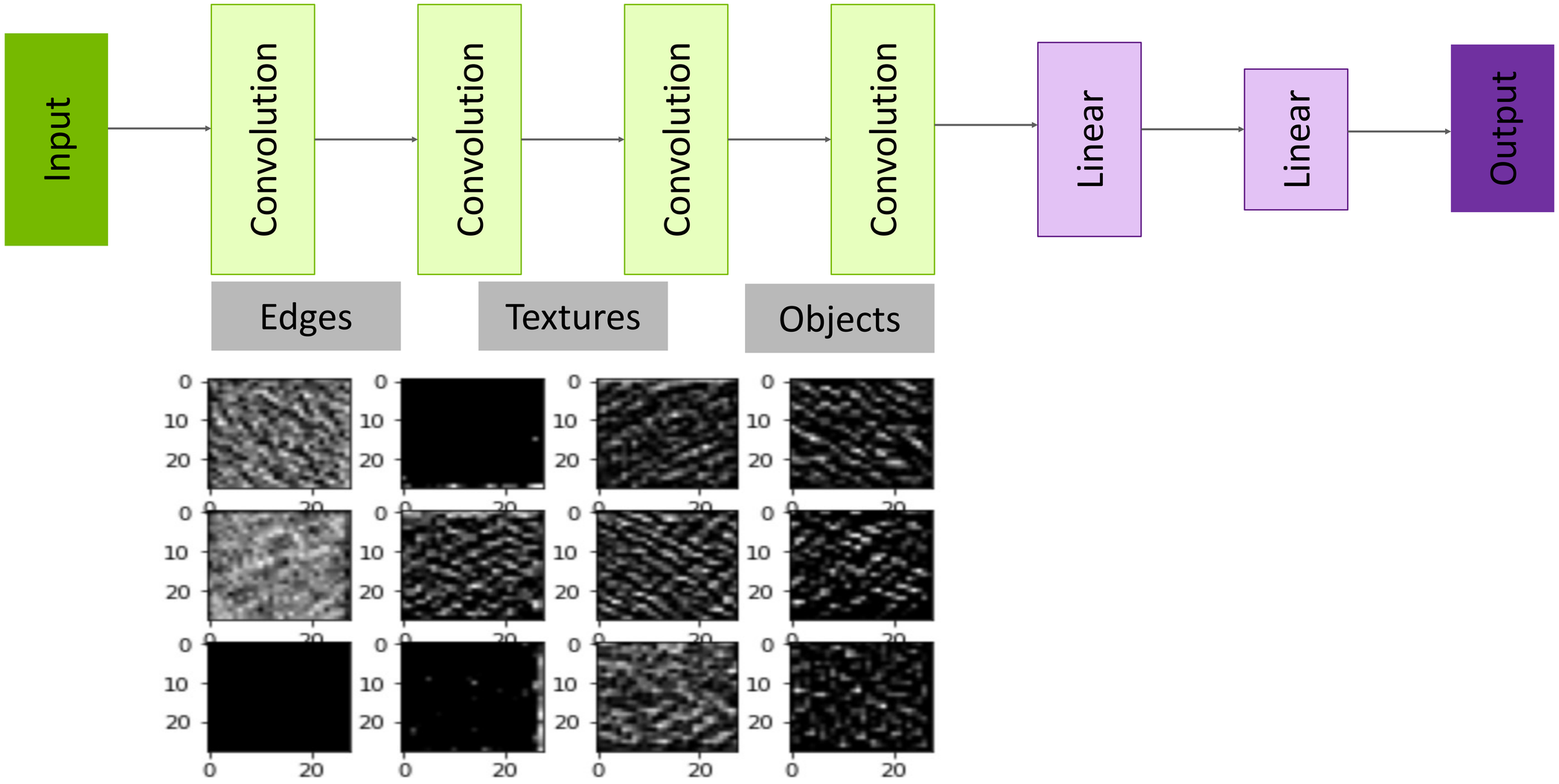
0	0	0
0	1	0
0	0	0

Horizontal Edges



1	2	1
0	0	0
-1	-2	-1

Neural Network Perception

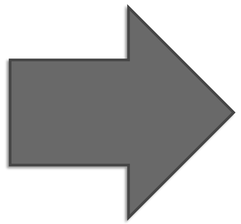


Neural Network Perception

Other Layers in the Model

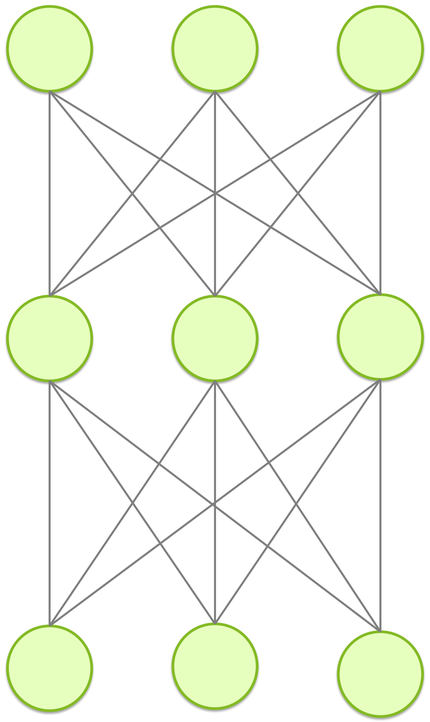
Max Pooling

110	256	153	67
12	89	88	43
10	15	50	55
23	9	49	23

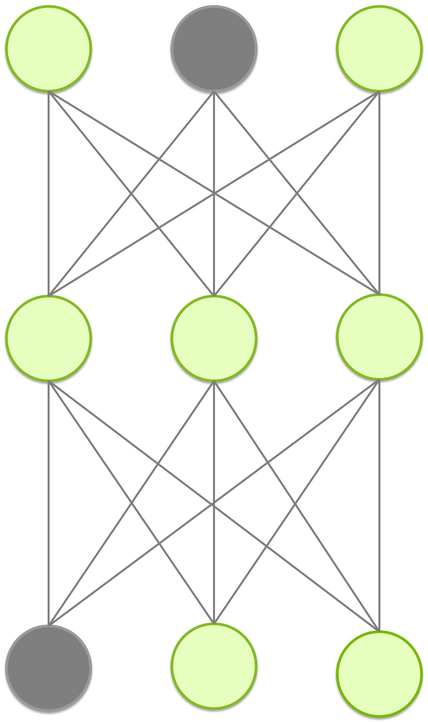


256	153
23	55

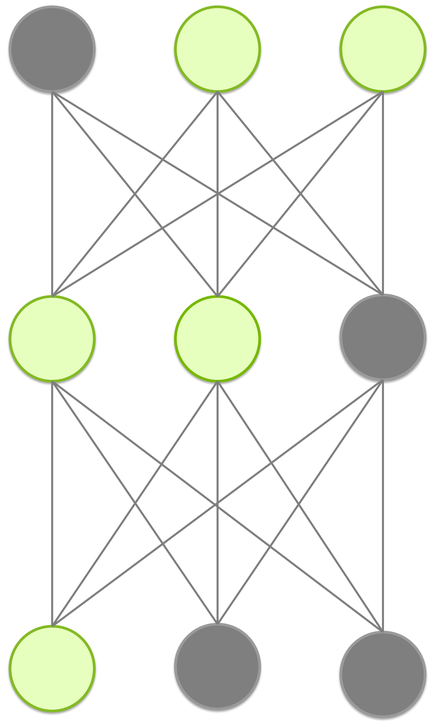
Dropout



rate = 0

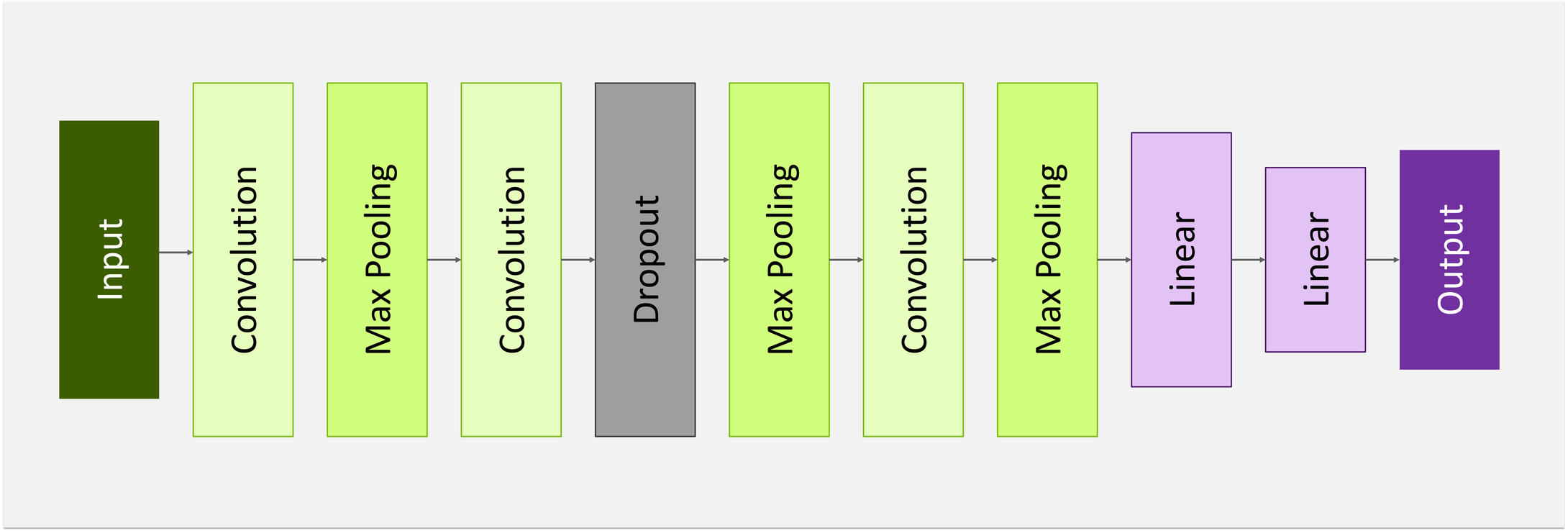


rate = .2



rate = .4

Whole Architecture



Let’s go!

Fundamentals of Deep Learning

Part 4: Data Augmentation and Deployment

Agenda

- Part 1: An Introduction to Deep Learning
- Part 2: How a Neural Network Trains
- Part 3: Convolutional Neural Networks
- Part 4: Data Augmentation and Deployment
- Part 5: Pre-Trained Models
- Part 6: Advanced Architectures

Recap of the Exercise

Analysis

- CNN increased validation accuracy
- Still seeing training accuracy higher than validation

Recap of the Exercise

Analysis

- CNN increased validation accuracy
- Still seeing training accuracy higher than validation

Solution

- Clean data provides better examples
- Dataset variety helps the model generalize



Data Augmentation

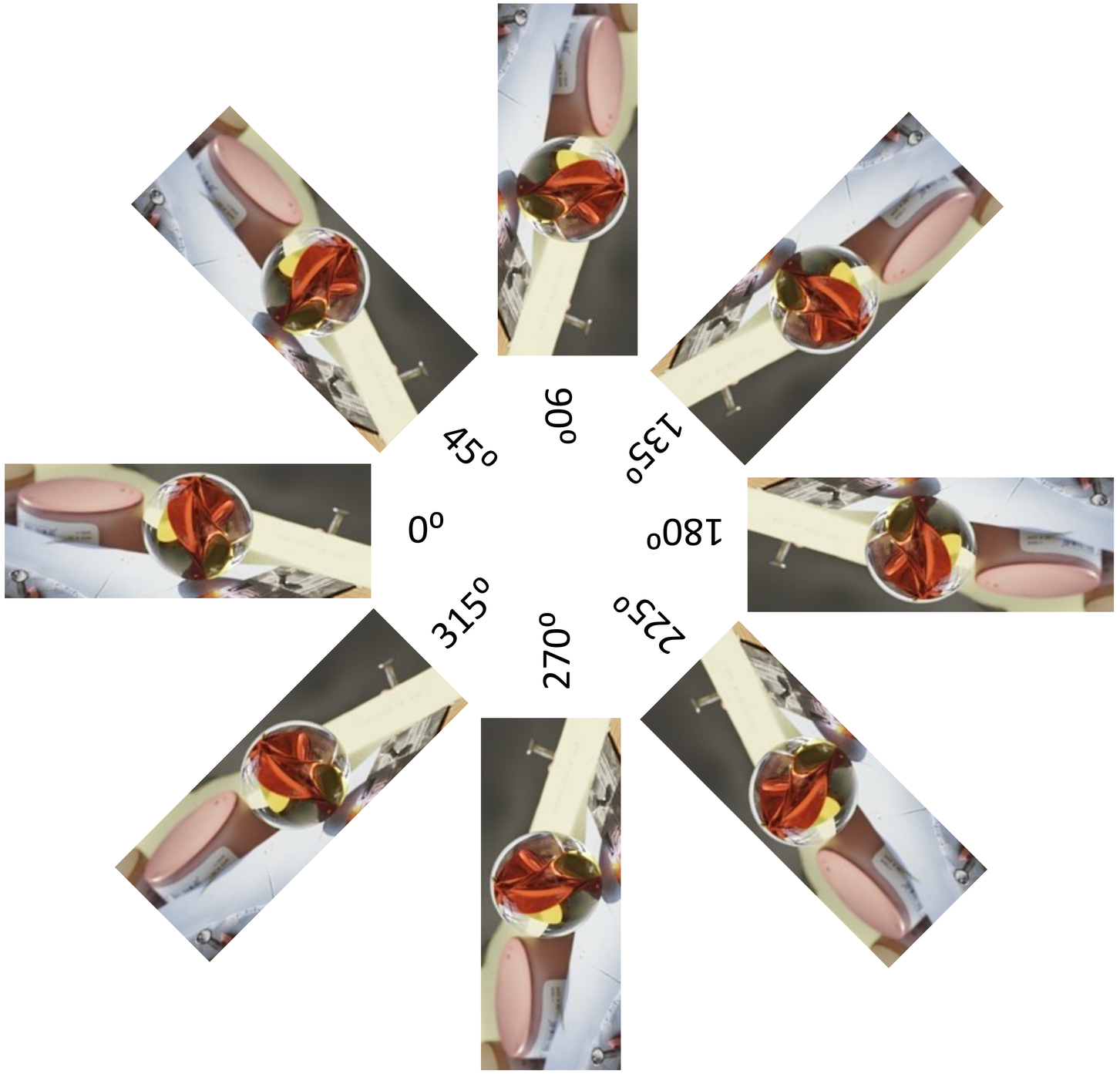
Data Augmentation

Image Flipping

Horizontal Flip

Vertical Flip

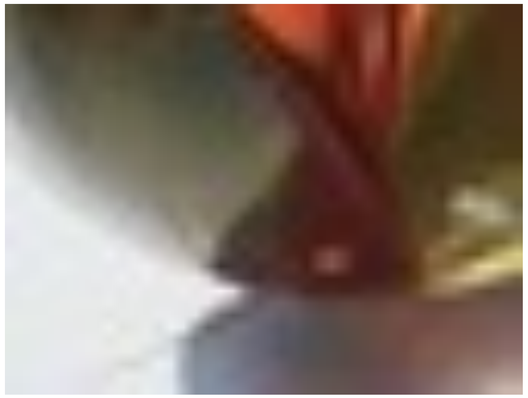
Rotation

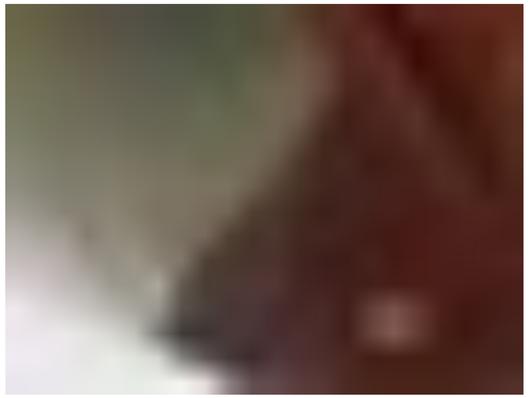


Zooming



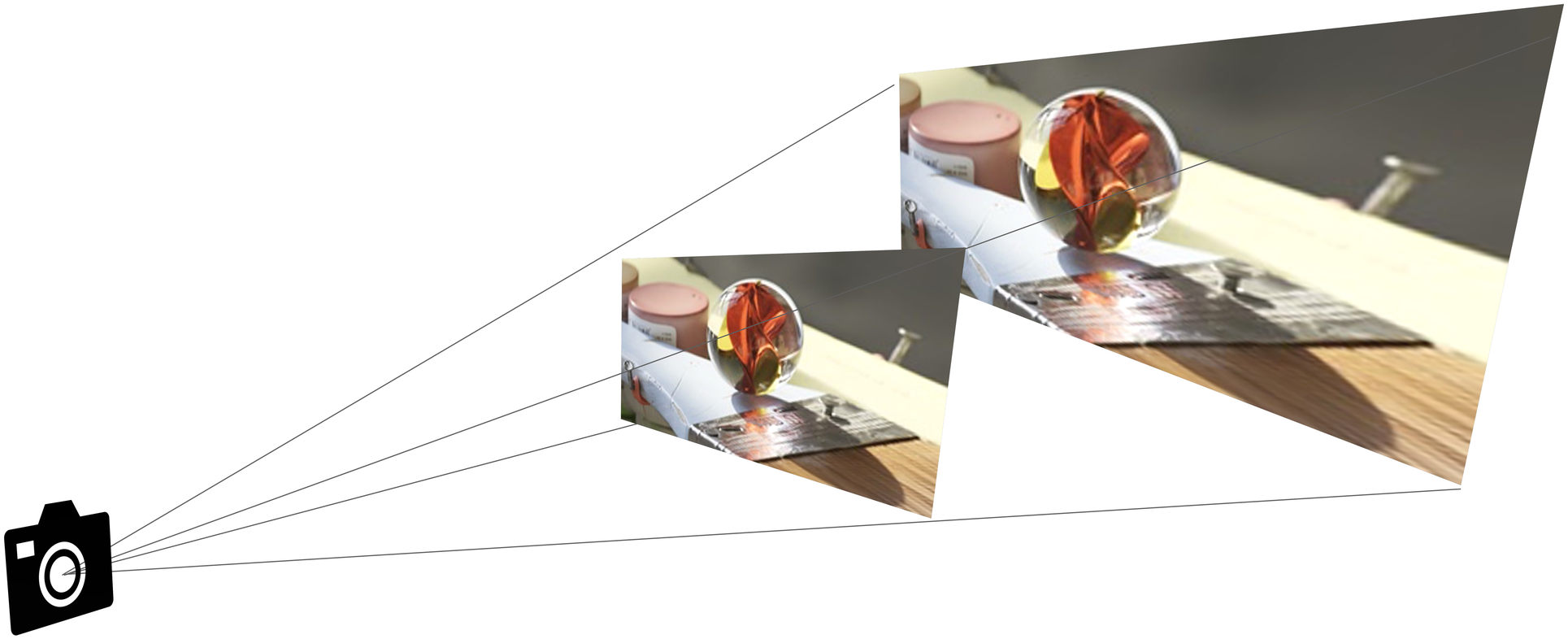




Width And Height Shifting

Homography

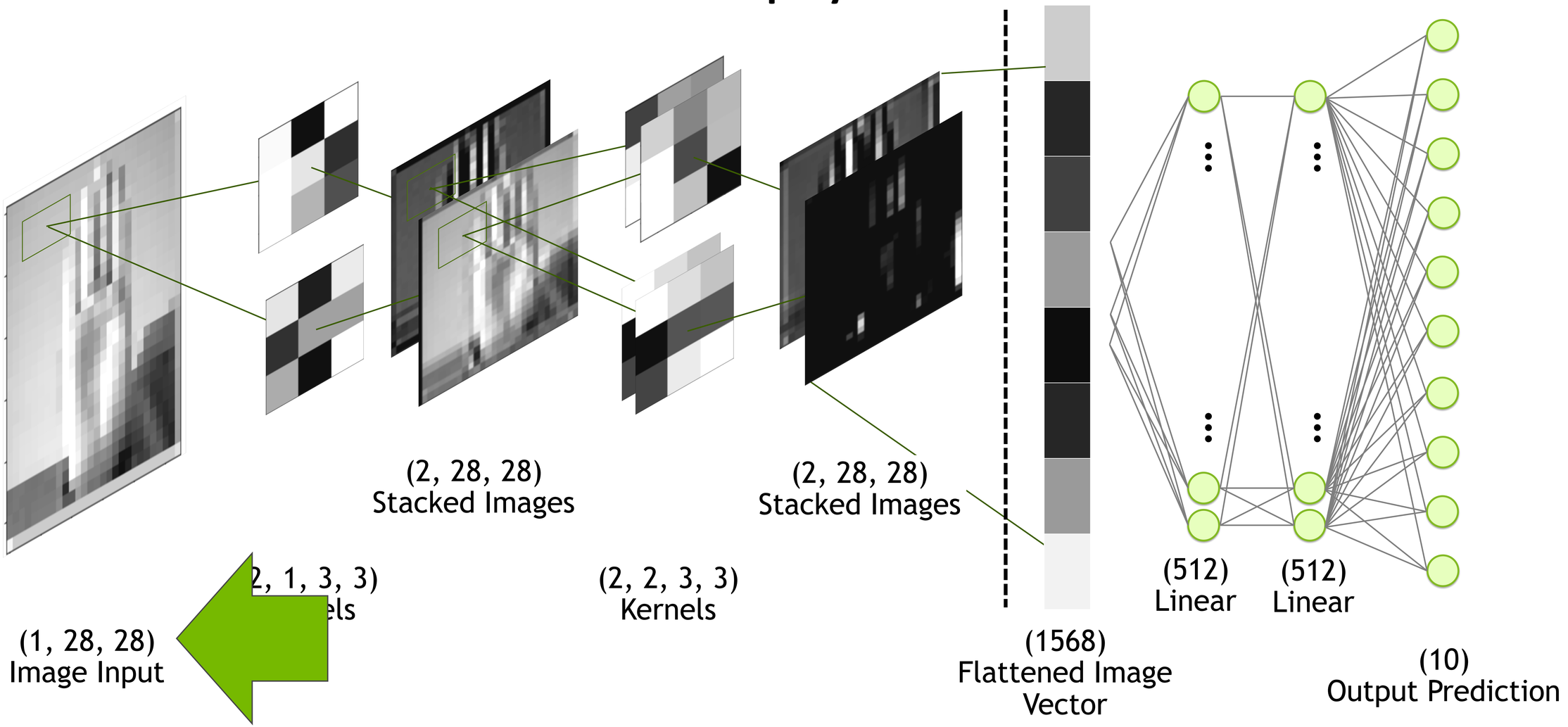


Brightness

Channel Shifting

Model Deployment

Model Deployment



Model Deployment

Training Batch Input

Convolution

Max Pooling

...

Model Deployment

(3, 287, 433)

Resize

(3, 220, 155)

Greyscale

(1, 220, 155)

“Batch”

(1, 1, 220, 155)

Let’s Try It Out!

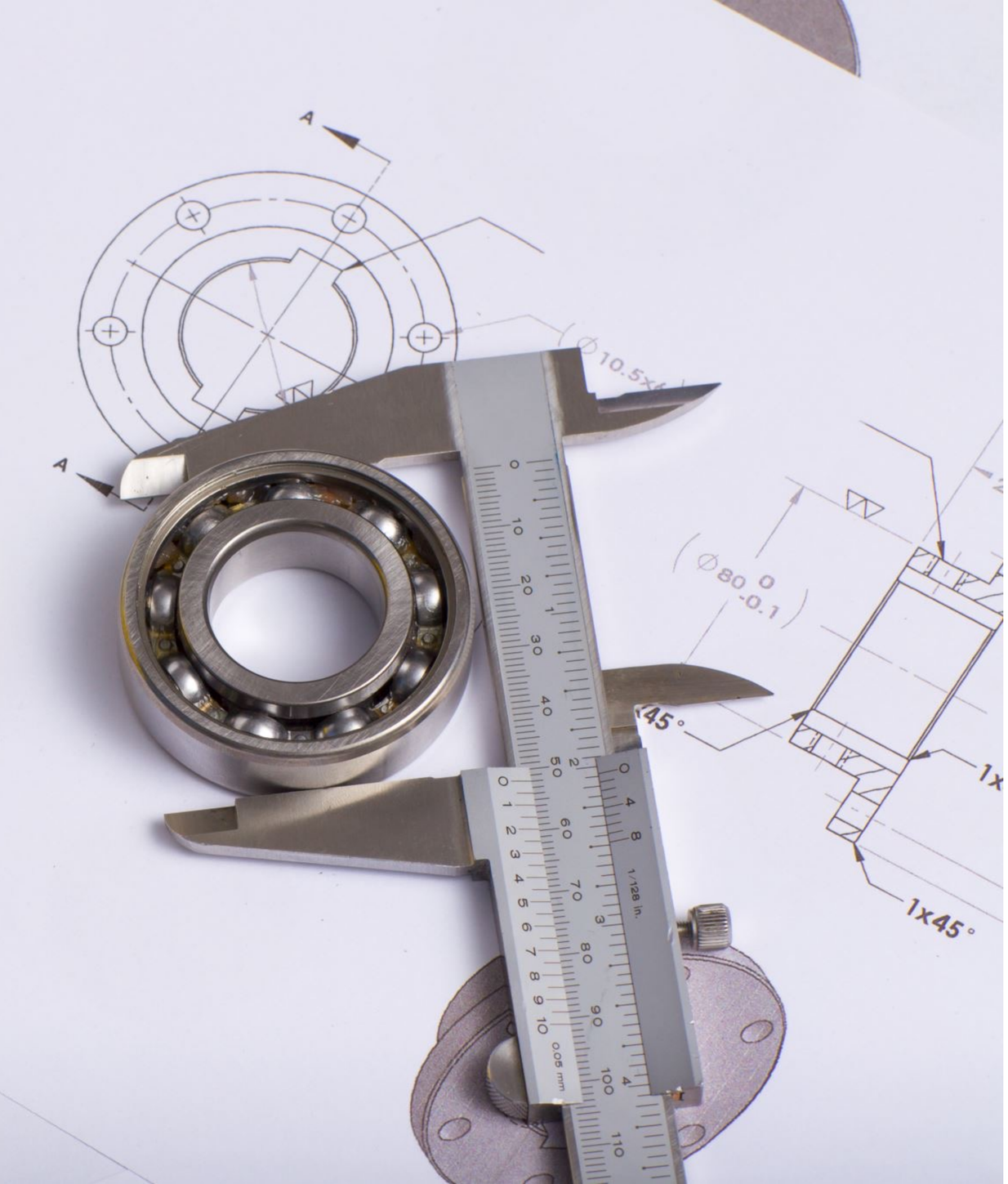
Fundamentals of Deep Learning

Part 5: Pre-trained Models

Agenda

- Part 1: An Introduction to Deep Learning
- Part 2: How a Neural Network Trains
- Part 3: Convolutional Neural Networks
- Part 4: Data Augmentation and Deployment
- Part 5: Pre-Trained Models
- Part 6: Advanced Architectures

Review So Far



Review So Far

- Learning Rate
- Number of Layers
- Neurons per Layer
- Activation Functions
- Dropout
- Data

Pre-Trained Models

Pre-Trained Models

TensorFlow Hub

Keras

PYTORCH HUB

Pre-Trained Models

VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

Karen Simonyan* **& Andrew Zisserman+**
Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk

The Next Challenge

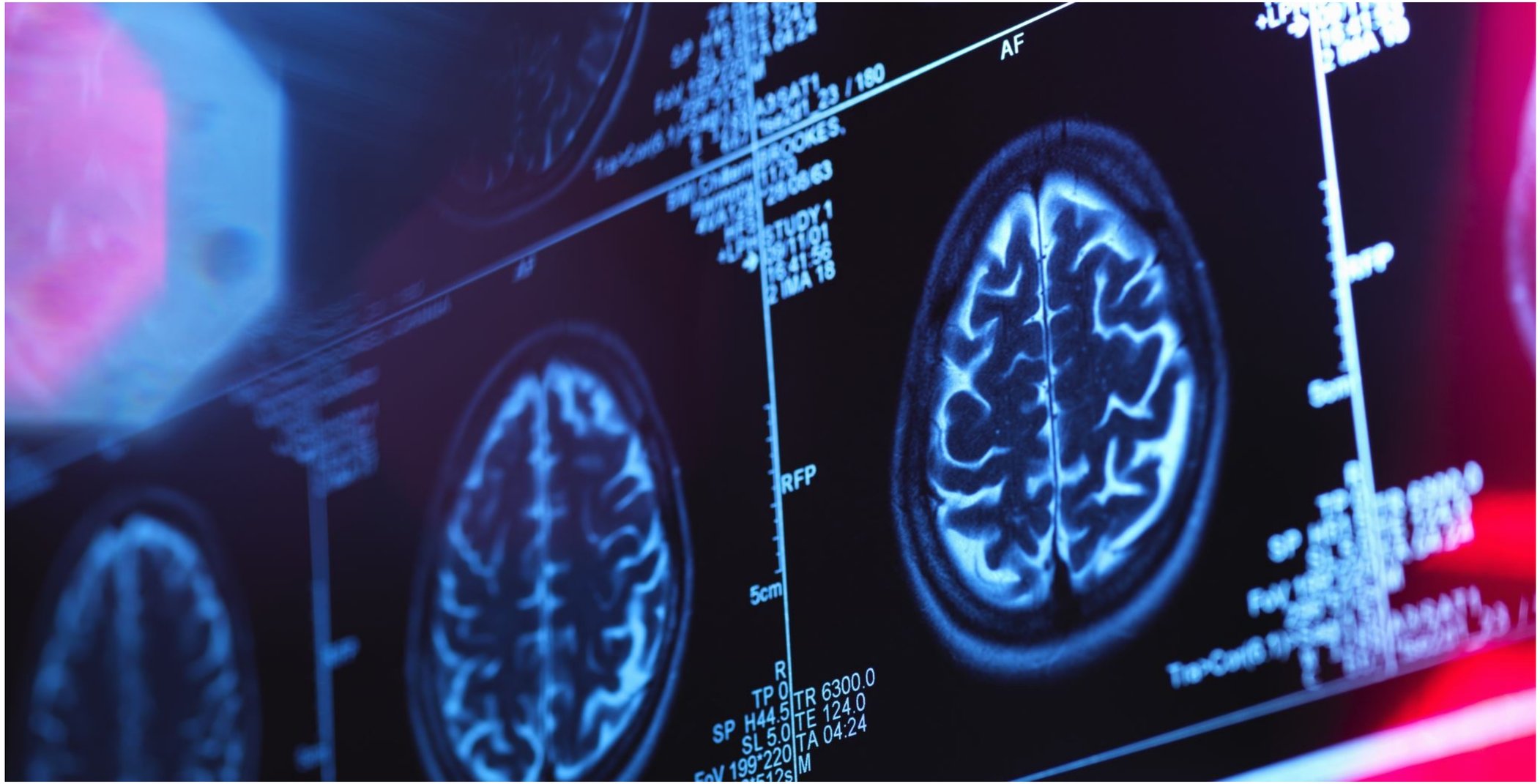
An Automated Doggy Door

Transfer Learning

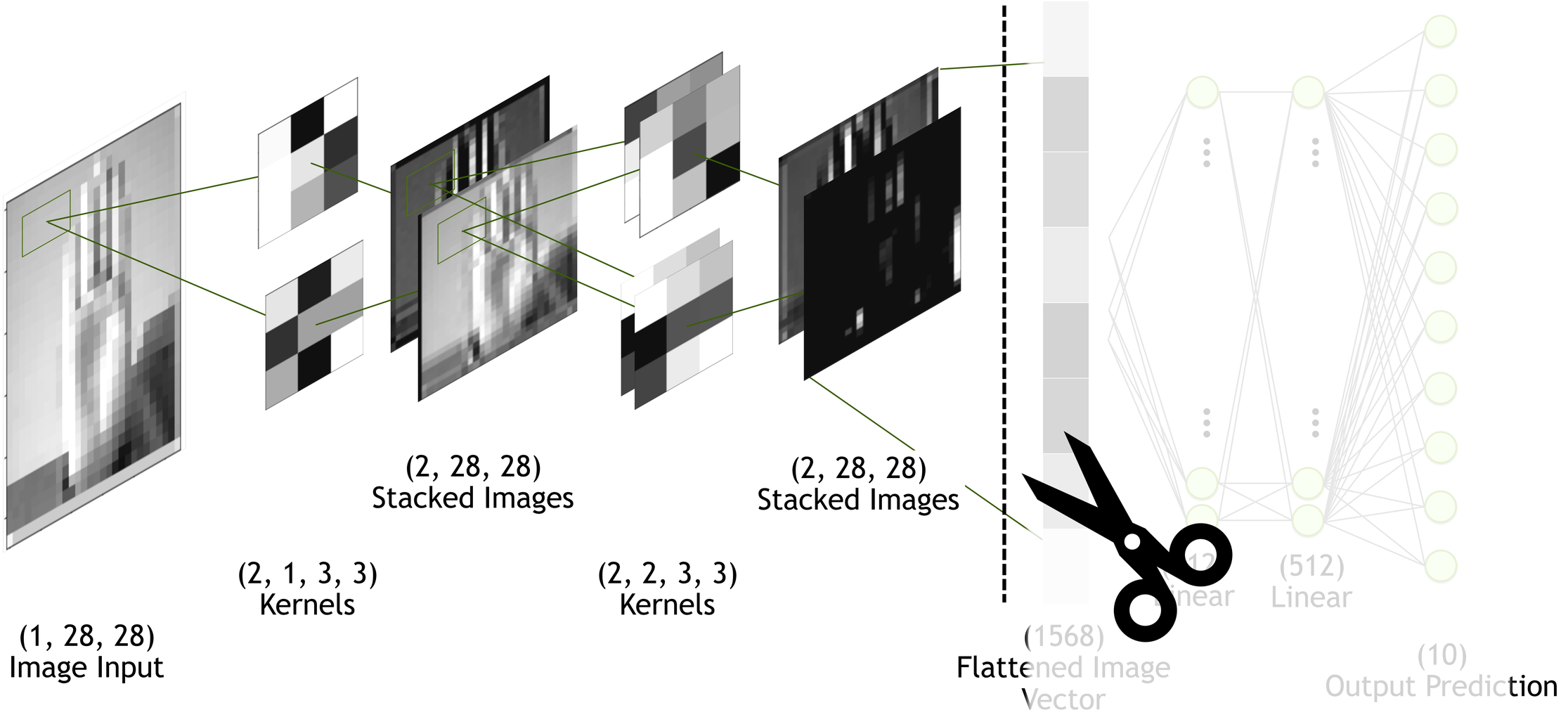
The Challenge After

An Automated Presidential Doggy Door

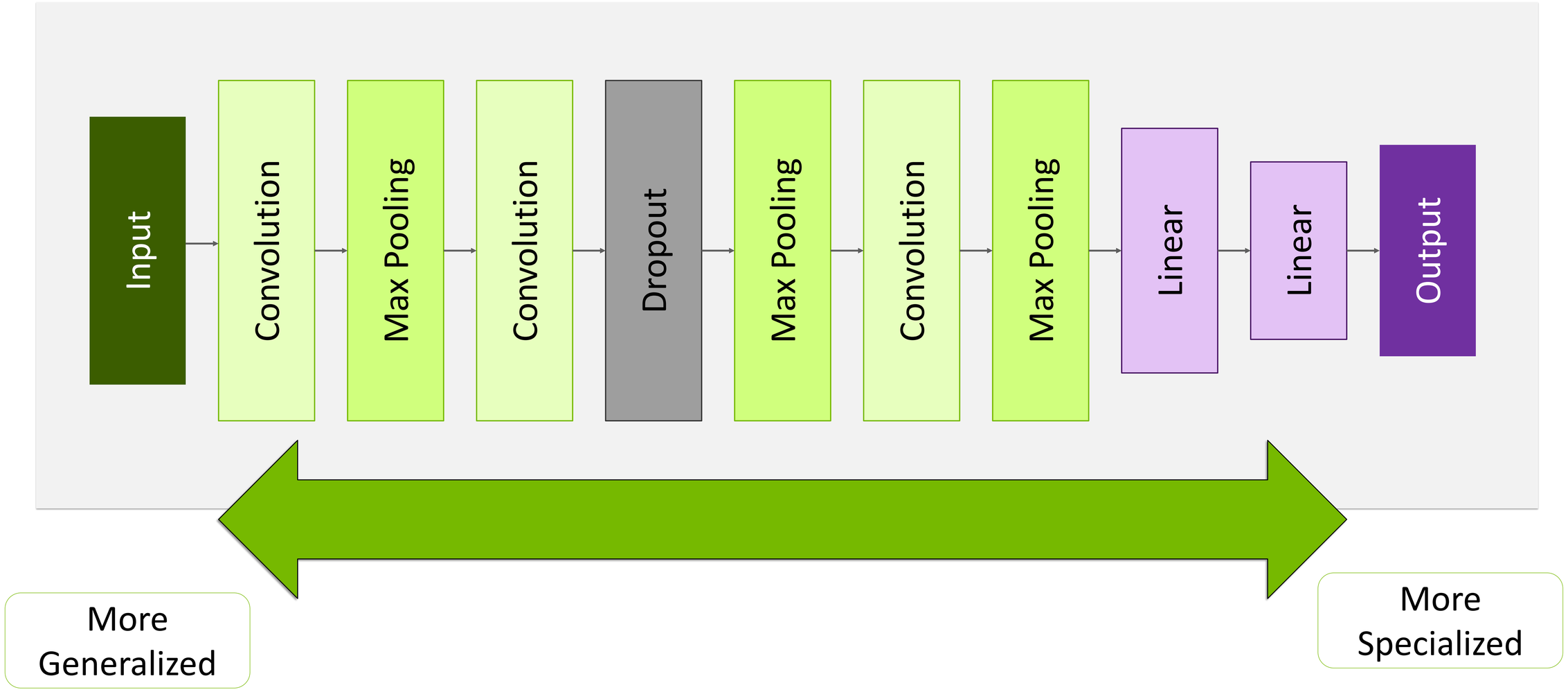
Transfer Learning



Transfer Learning



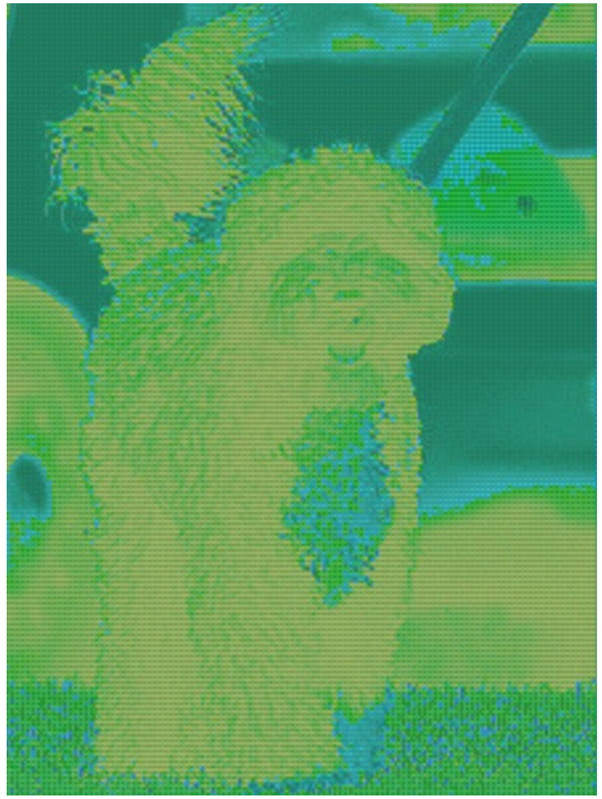
Transfer Learning



Transfer Learning

Freezing the Model?

Transfer Learning



Let’s Get Started!

Fundamentals of Deep Learning

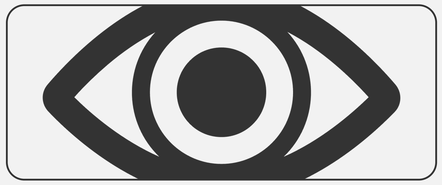
Part 6: Advanced Architectures

Agenda

- Part 1: An Introduction to Deep Learning
- Part 2: How a Neural Network Trains
- Part 3: Convolutional Neural Networks
- Part 4: Data Augmentation and Deployment
- Part 5: Pre-Trained Models
- Part 6: Advanced Architectures

Moving Forward

Fields of AI

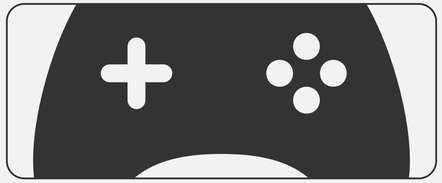


Computer Vision

- Optometry

Natural Language Processing

- Linguistics



Reinforcement Learning

- Game Theory
- Psychology

Anomaly Detection

- Security
- Medicine

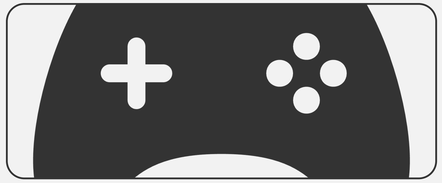
Fields of AI

Computer Vision

- Optometry

Natural Language Processing

- Linguistics



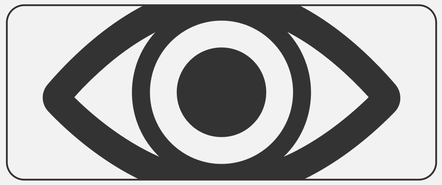
Reinforcement Learning

- Game Theory
- Psychology

Anomaly Detection

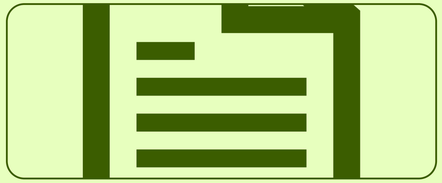
- Security
- Medicine

Fields of AI



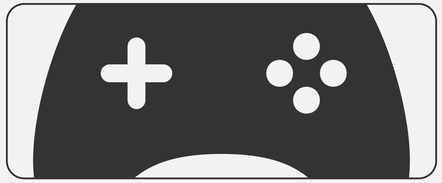
Computer Vision

- Optometry



Natural Language Processing

- Linguistics



Reinforcement Learning

- Game Theory
- Psychology

Anomaly Detection

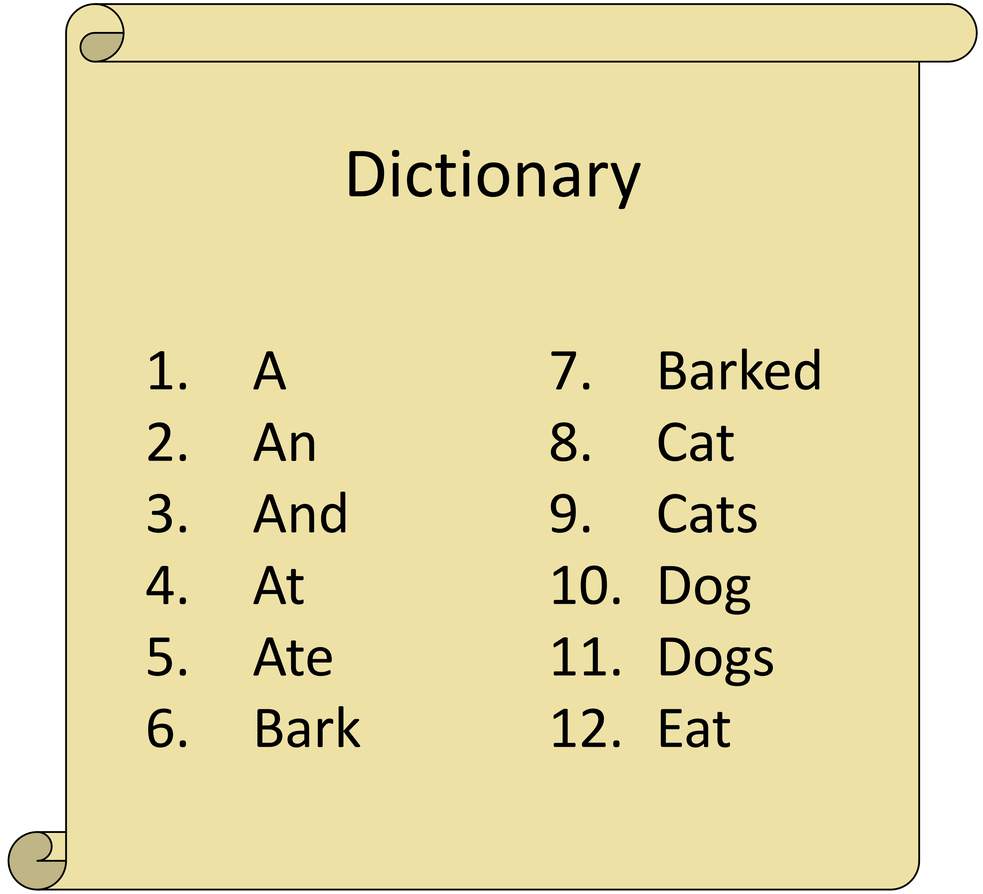
- Security
- Medicine

Natural Language Processing

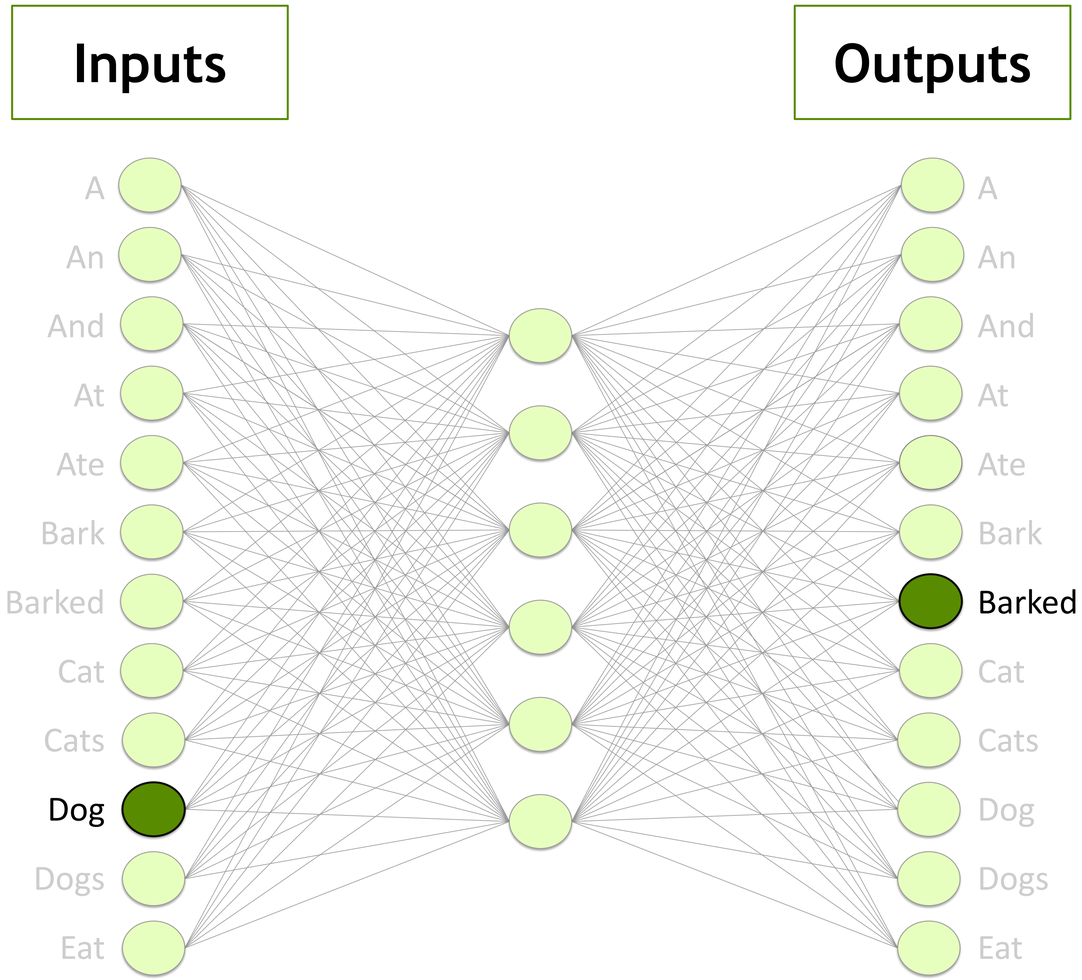
From Words to Numbers

“A dog barked at a cat.”

[1, 10, 7, 4, 1, 8]



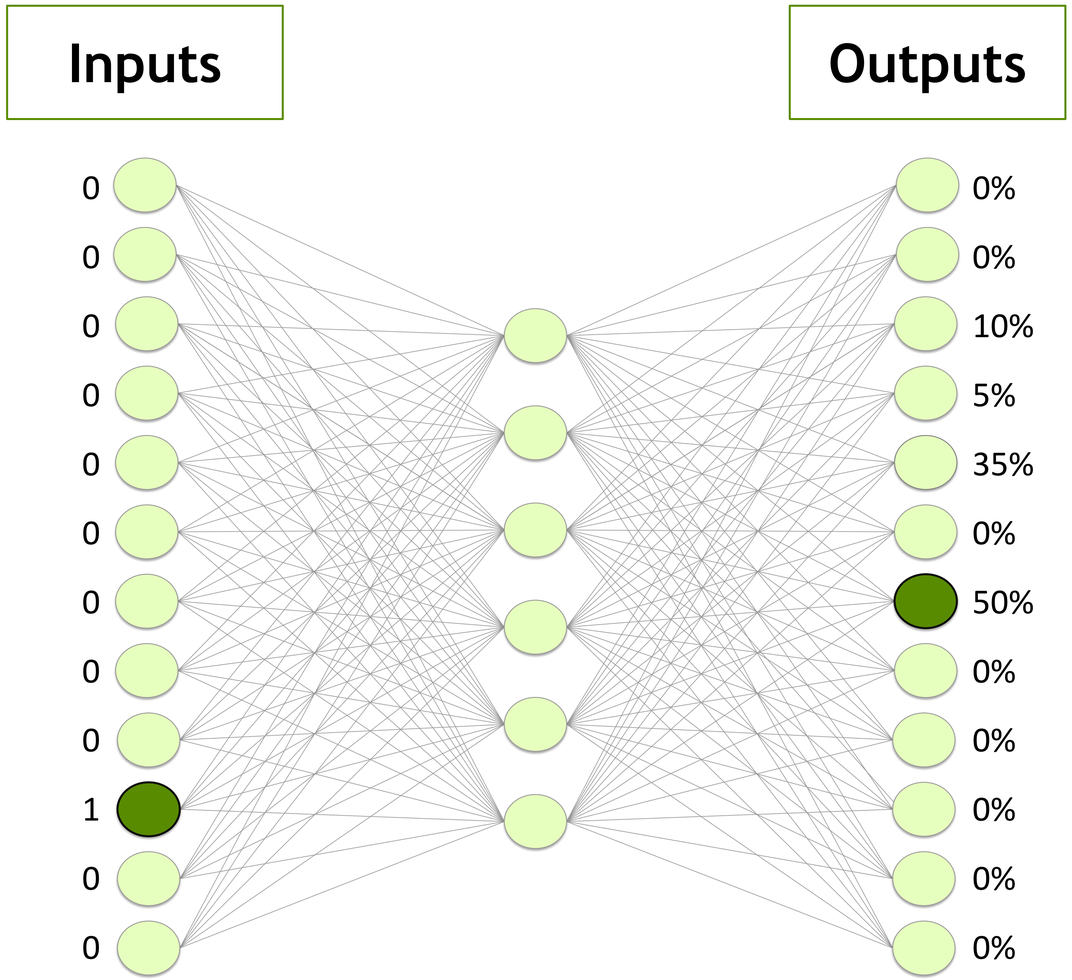
From Words to Numbers



Dictionary

1. A
2. An
3. And
4. At
5. Ate
6. Bark
7. Barked
8. Cat
9. Cats
10. Dog
11. Dogs
12. Eat

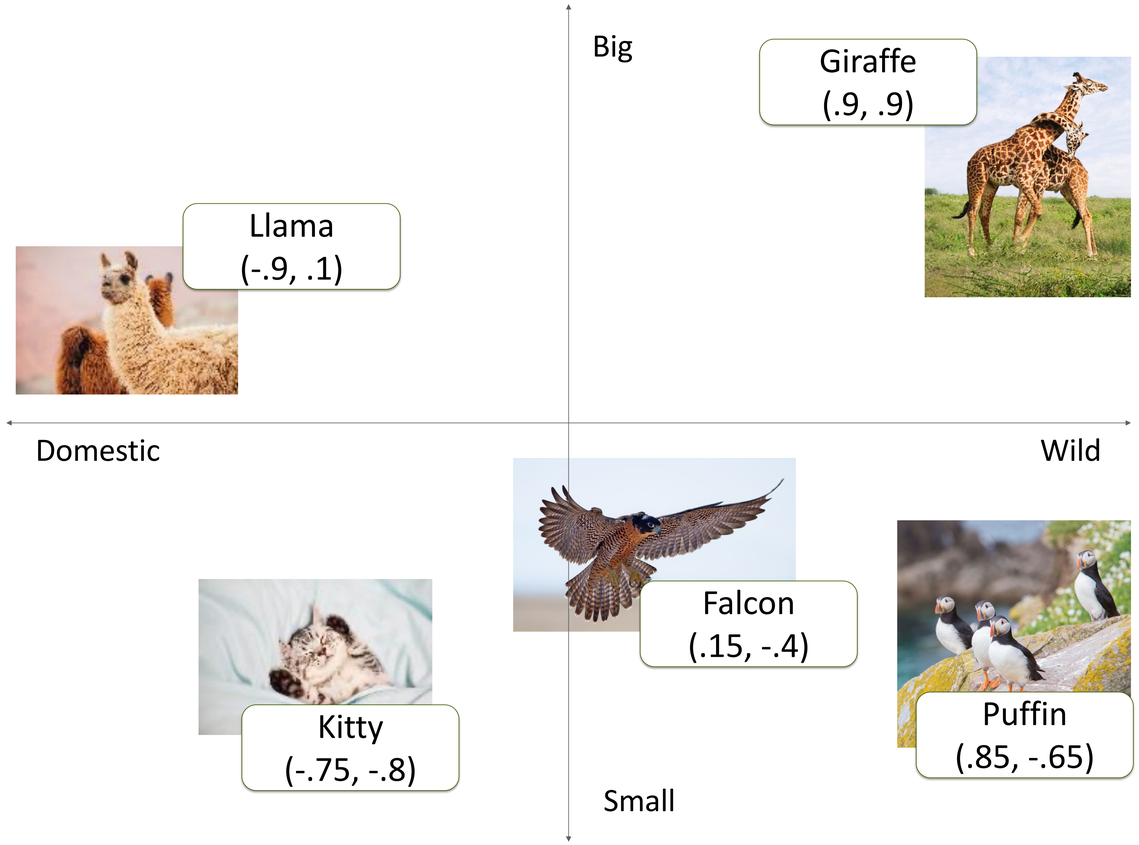
From Words to Numbers



Dictionary

1. A
2. An
3. And
4. At
5. Ate
6. Bark
7. Barked
8. Cat
9. Cats
10. Dog
11. Dogs
12. Eat

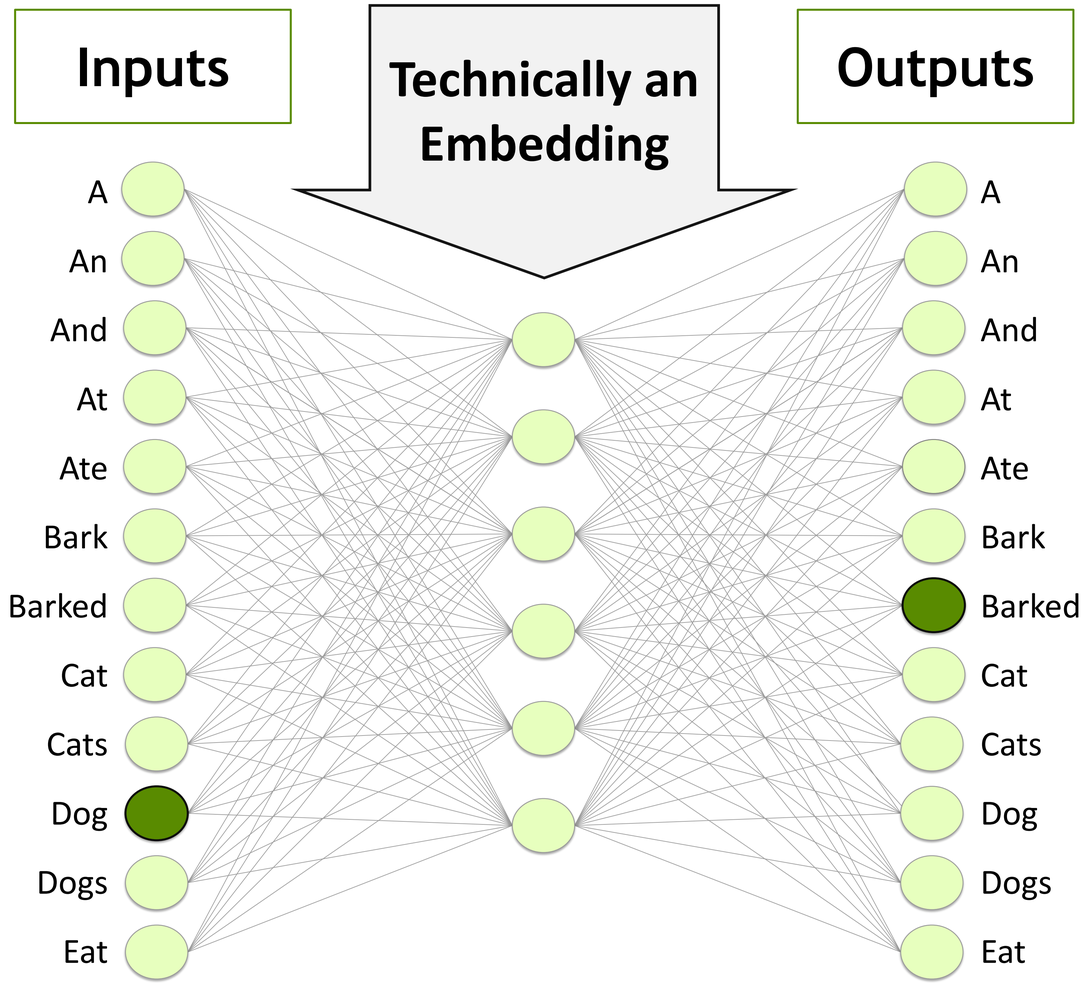
From Words to Numbers



Bigger Dictionary

1.	A	33.	Barked	65.	Eaten
2.	An	34.	Cat	66.	A
3.	And	35.	Cats	67.	An
4.	At	36.	Dog	68.	And
5.	Ate	37.	Dogs	69.	At
6.	Bark	38.	Eat	70.	Ate
7.	Barked	39.	Eaten	71.	Bark
8.	Cat	40.	A	72.	Barked
9.	Cats	41.	An	73.	Cat
10.	Dog	42.	And	74.	Cats
11.	Dogs	43.	At	75.	Dog
12.	Eat	44.	Ate	76.	Dogs
13.	Eaten	45.	Bark	77.	Eat
14.	A	46.	Barked	78.	Eaten
15.	An	47.	Cat	79.	…
16.	And	48.	Cats	80.	…
17.	At	49.	Dog	81.	…
18.	Ate	50.	Dogs	82.	…
19.	Bark	51.	Eat		
20.	Barked	52.	Eaten		
21.	Cat	53.	A		
22.	Cats	54.	An		
23.	Dog	55.	And		
24.	Dogs	56.	At		
25.	Eat	57.	Ate		
26.	Eaten	58.	Bark		
27.	A	59.	Barked		
28.	An	60.	Cat		
29.	And	61.	Cats		
30.	At	62.	Dog		
31.	Ate	63.	Dogs		
32.	Bark	64.	Eat		

From Words to Numbers



Bigger Dictionary

1.	A	33.	Barked	65.	Eaten
2.	An	34.	Cat	66.	A
3.	And	35.	Cats	67.	An
4.	At	36.	Dog	68.	And
5.	Ate	37.	Dogs	69.	At
6.	Bark	38.	Eat	70.	Ate
7.	Barked	39.	Eaten	71.	Bark
8.	Cat	40.	A	72.	Barked
9.	Cats	41.	An	73.	Cat
10.	Dog	42.	And	74.	Cats
11.	Dogs	43.	At	75.	Dog
12.	Eat	44.	Ate	76.	Dogs
13.	Eaten	45.	Bark	77.	Eat
14.	A	46.	Barked	78.	Eaten
15.	An	47.	Cat	79.	…
16.	And	48.	Cats	80.	…
17.	At	49.	Dog	81.	…
18.	Ate	50.	Dogs	82.	…
19.	Bark	51.	Eat		
20.	Barked	52.	Eaten		
21.	Cat	53.	A		
22.	Cats	54.	An		
23.	Dog	55.	And		
24.	Dogs	56.	At		
25.	Eat	57.	Ate		
26.	Eaten	58.	Bark		
27.	A	59.	Barked		
28.	An	60.	Cat		
29.	And	61.	Cats		
30.	At	62.	Dog		
31.	Ate	63.	Dogs		
32.	Bark	64.	Eat		

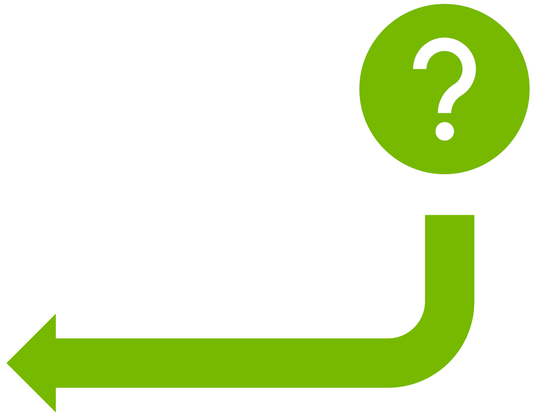
Attention

Sentence Prediction

I am the very model of a modern Major-Gineral,
I've information vegetable, animal, and mineral,

...

I'm very good at integral and differential calculus;
I know the scientific names of beings animalculous:
In short, in matters vegetable, animal, and mineral,
I am the very model of a m



~ Major-General Stanley

Sentence Prediction

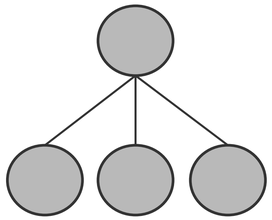
I am the very model of a modern Major-Gineral,
I've information vegetable, animal, and mineral,

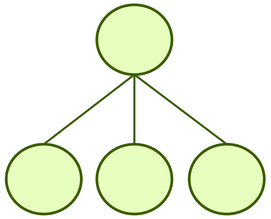
...

I'm very good at integral and differential calculus;
I know the scientific names of beings animalculous:
In short, in matters vegetable, animal, and mineral,
I am the very model of a modern Major-Gineral.

~ Major-General Stanley

Attention





I
am
the
very
model

5 x 3

Q

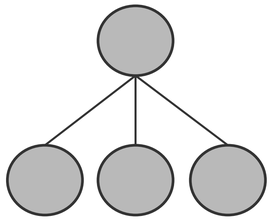
Query

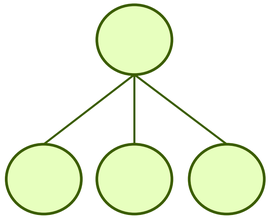
5 x 3

K

Key

Attention





I						
am						
the						
very						
model						

5 x 3

Q

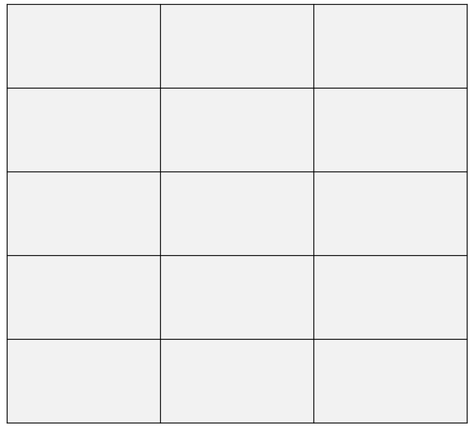
Query

5 x 3

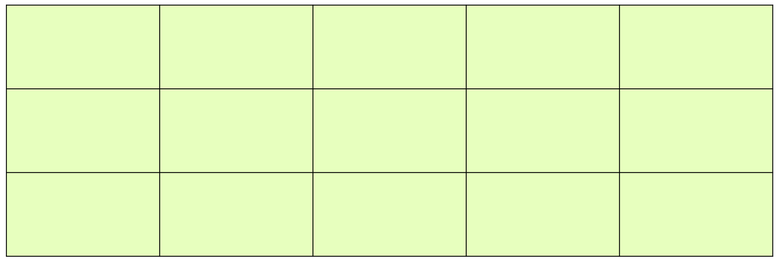
K

Key

Attention

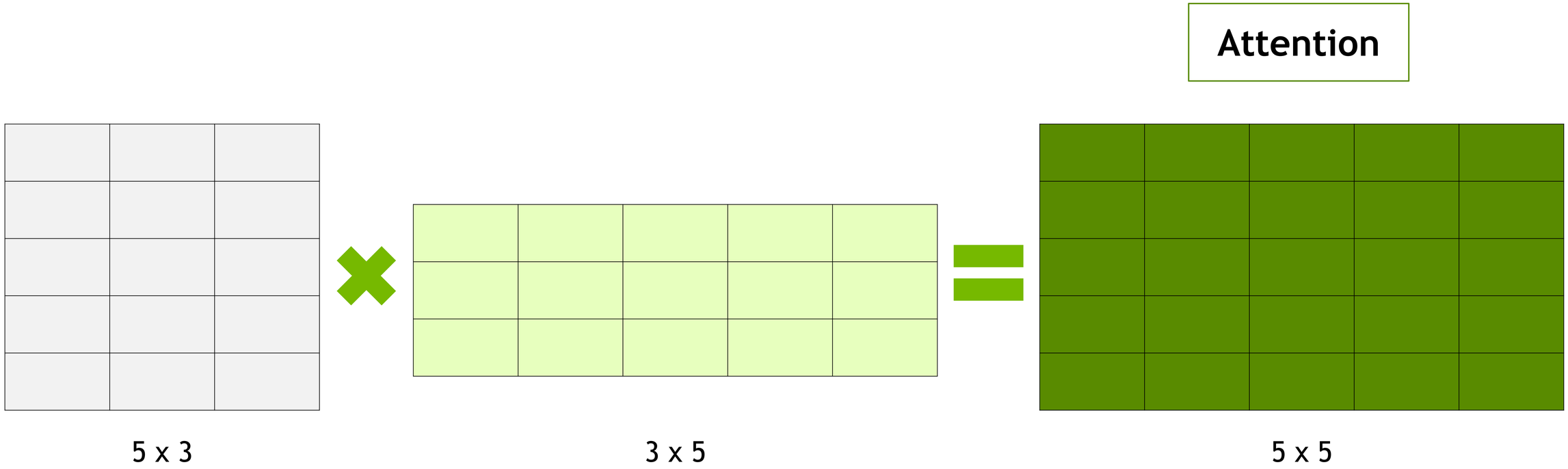


5 x 3

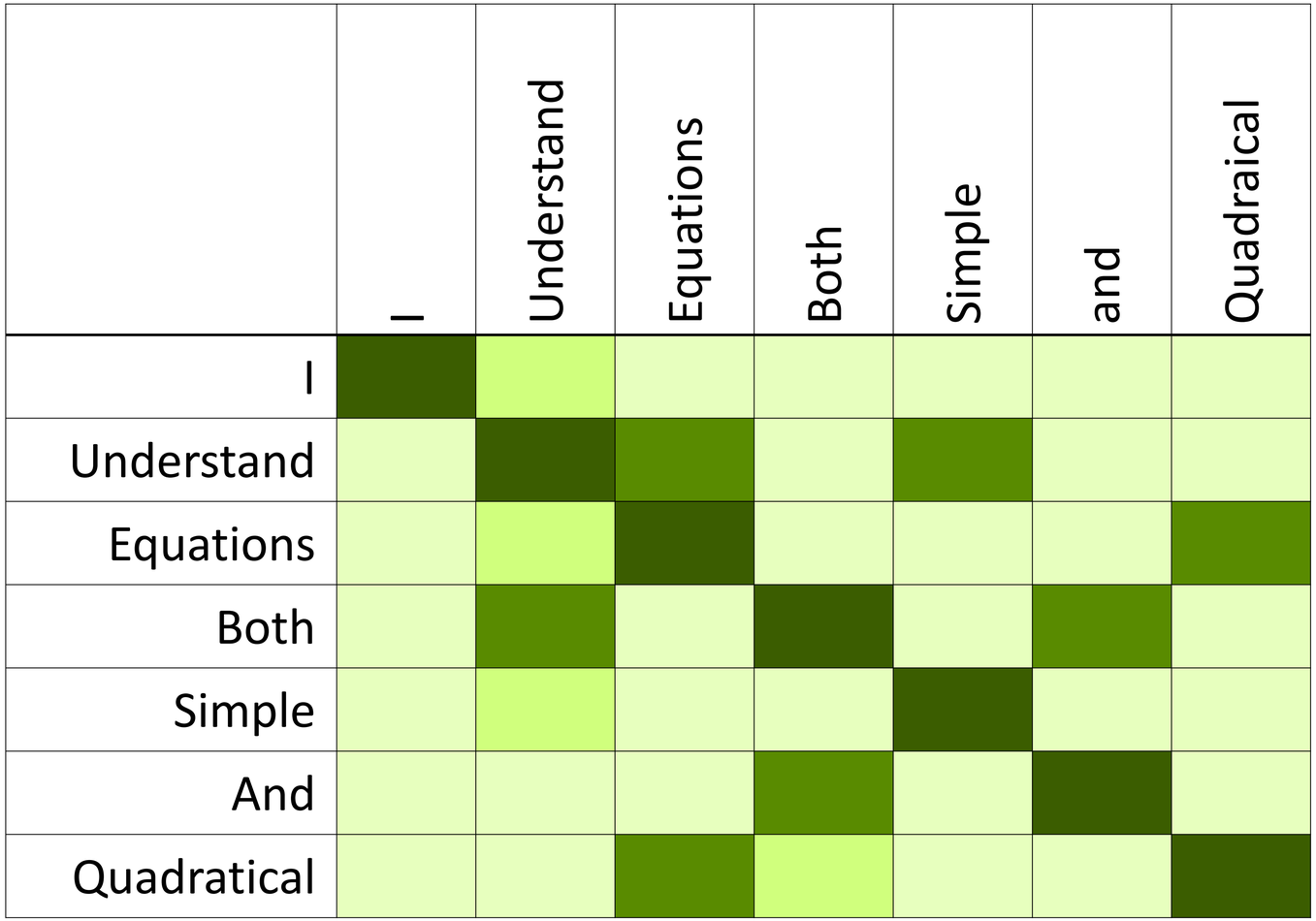


3 x 5

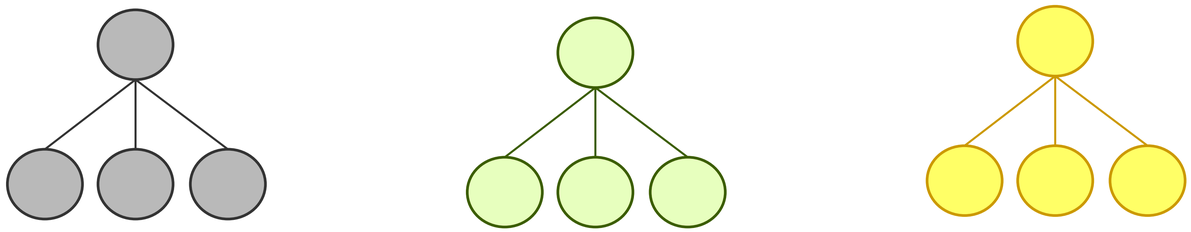
Attention



Attention


I
Understand
Equations
Both
Simple
and
Quadraical
I
Understand
Equations
Both
Simple
And
Quadratical

Attention



I									
am									
the									
very									
model									

5 x 3

Q

Query

5 x 3

K

Key

5 x 3

V

Value

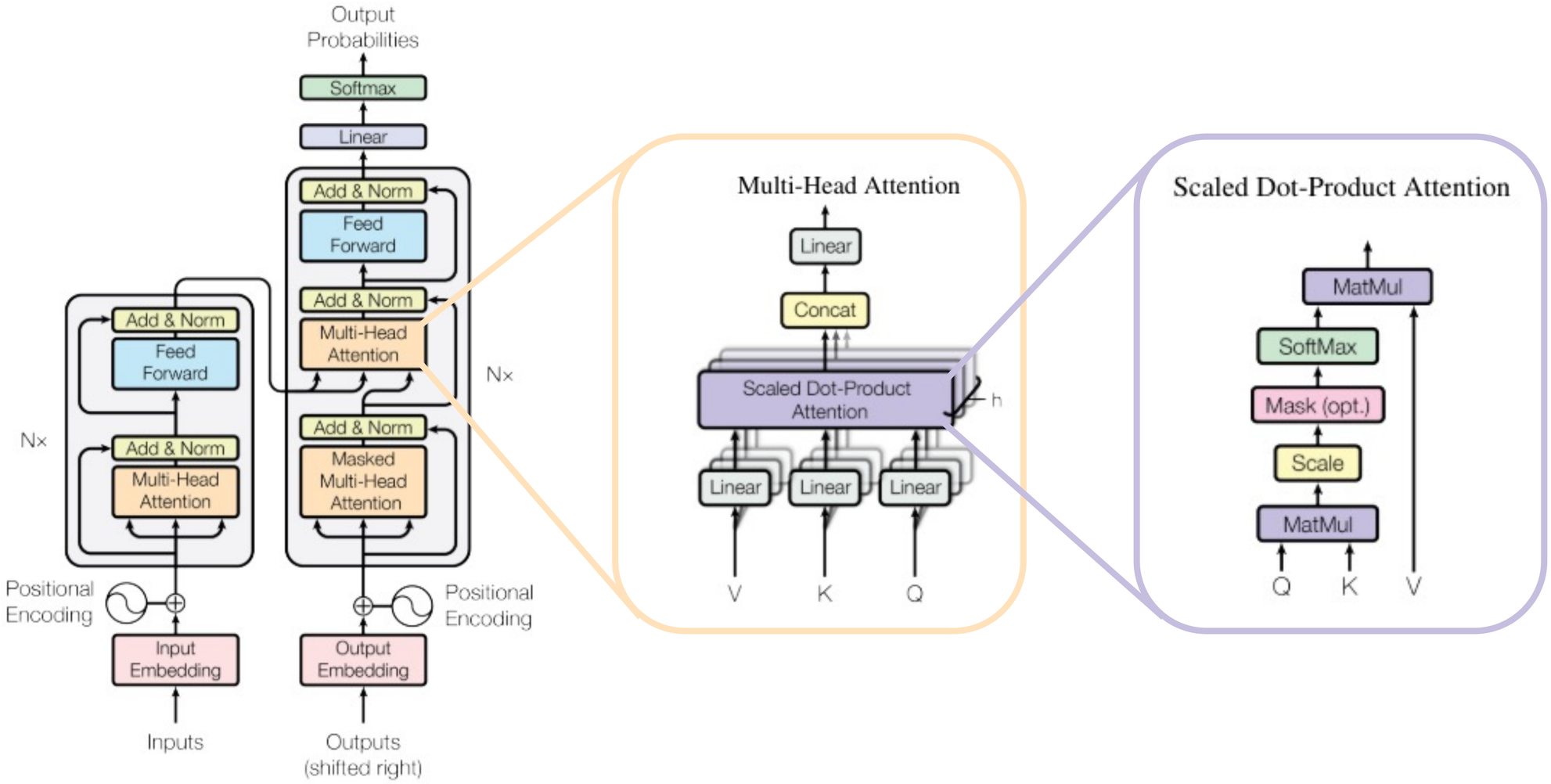
Attention

$$Z = softmax\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V$$

I												
am												
the												
very												
model												

5 x 3	5 x 3	5 x 3	5 x 3
Q	K	V	Z
Query	Key	Value	Attention Score

Transformers



BERT

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

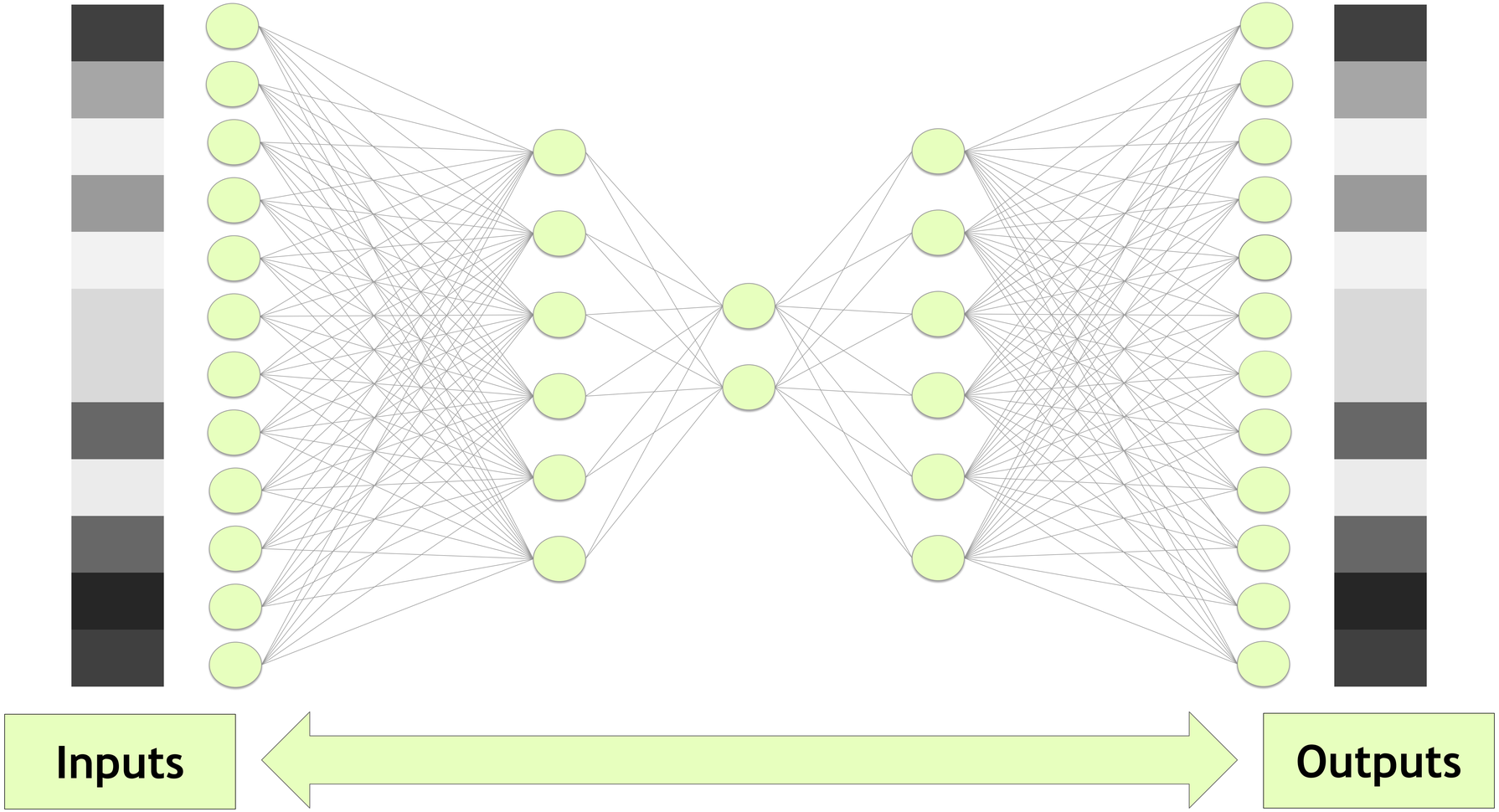
Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from

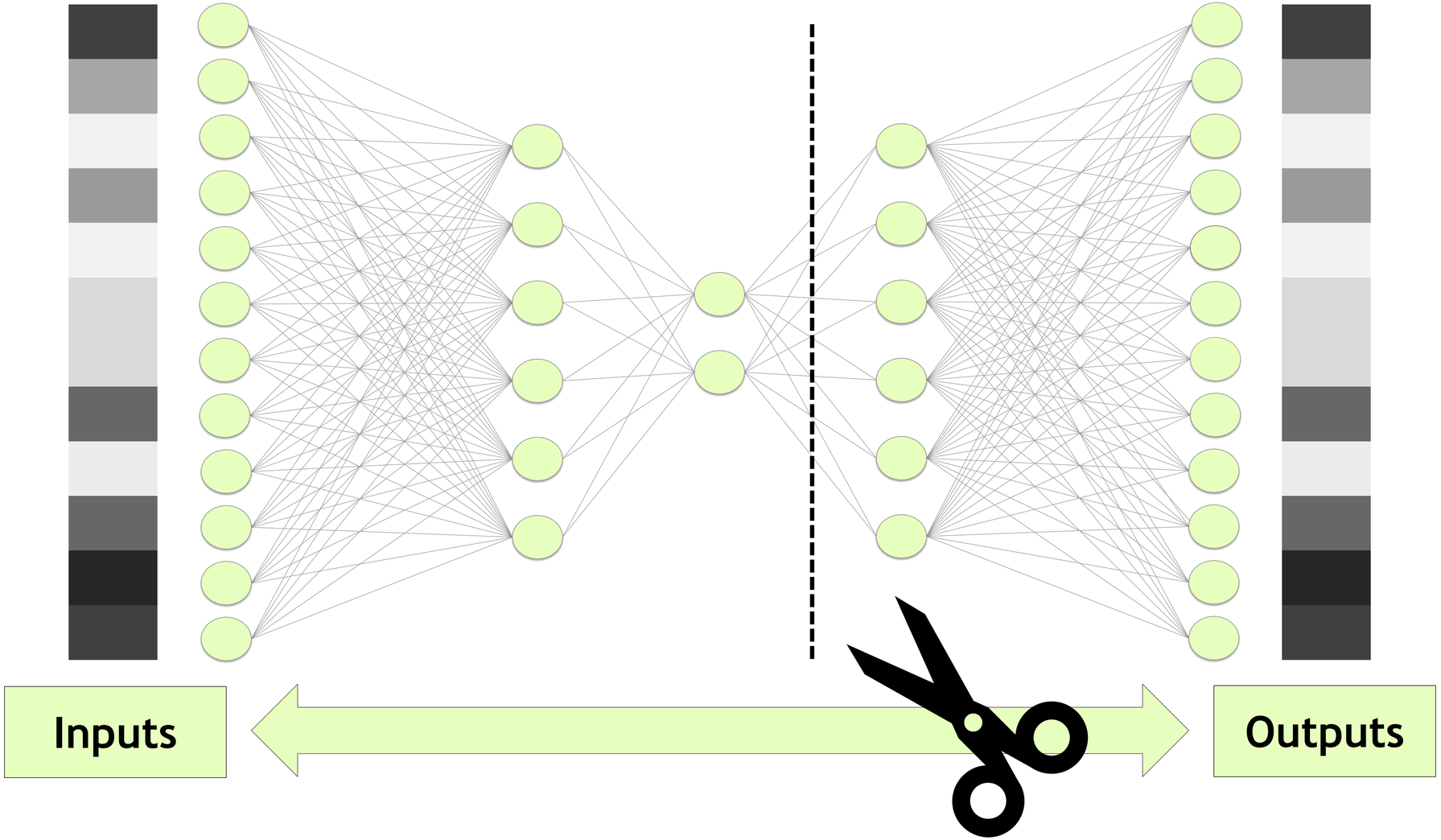
There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters

Other Architectures

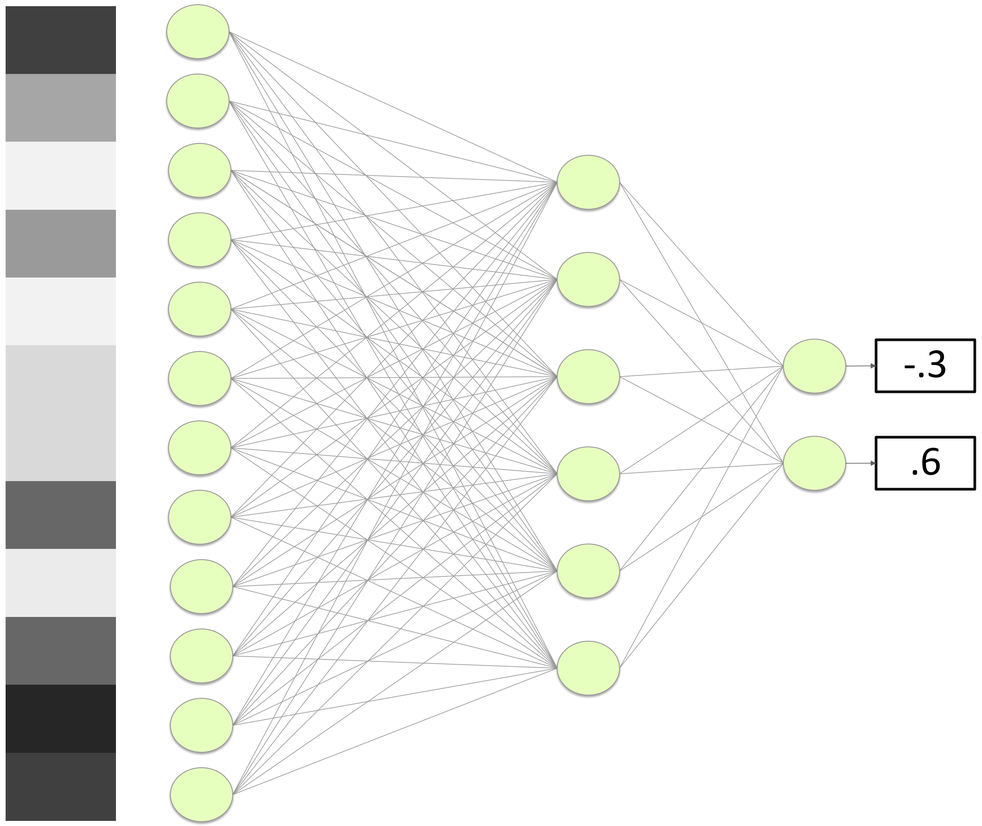
Autoencoders



Autoencoders

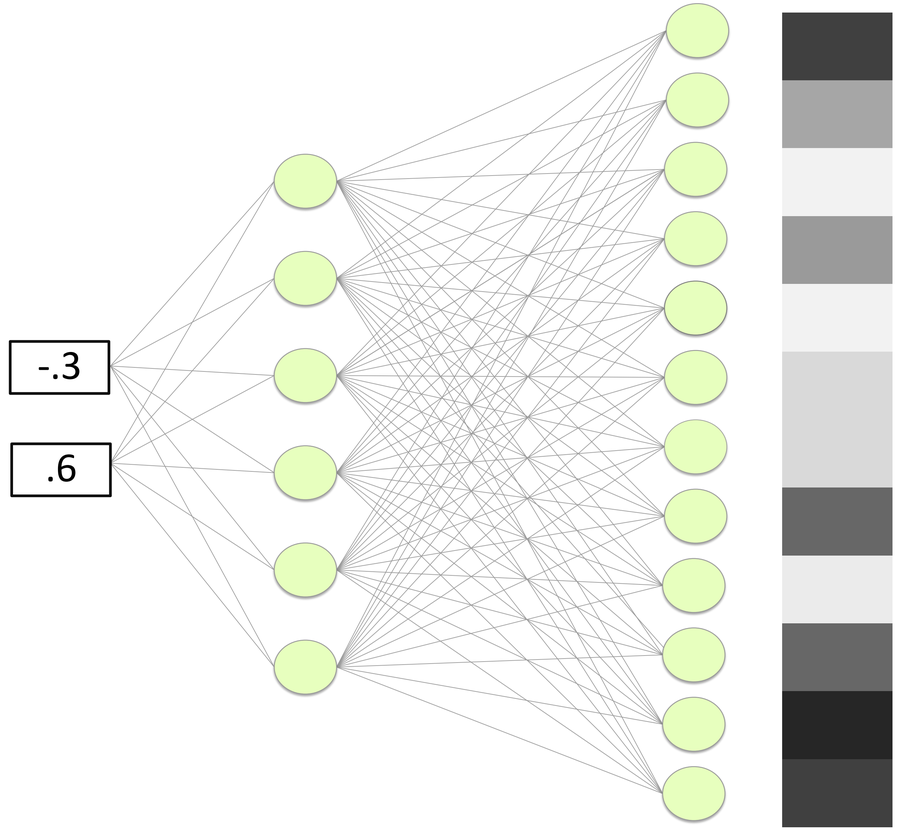


Autoencoders



-.3

.6



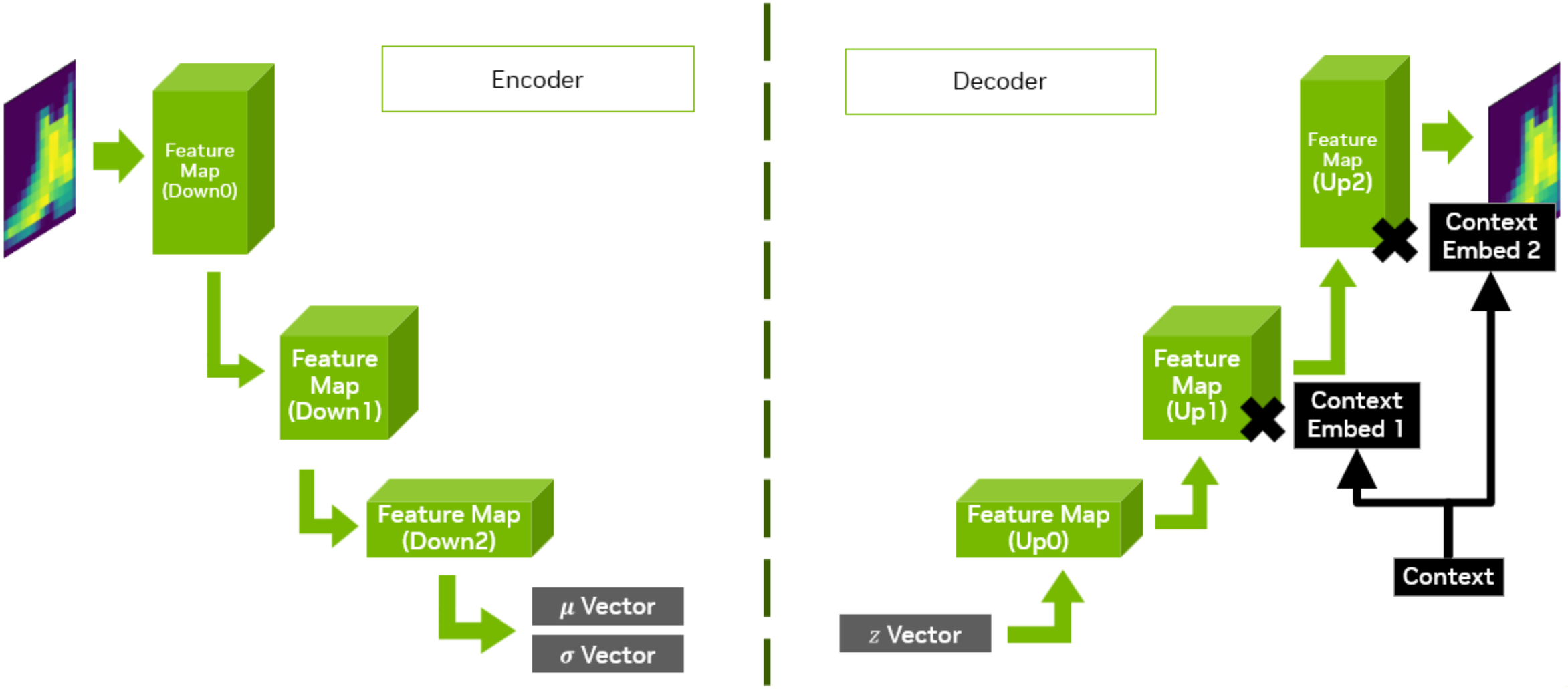
-.3

.6

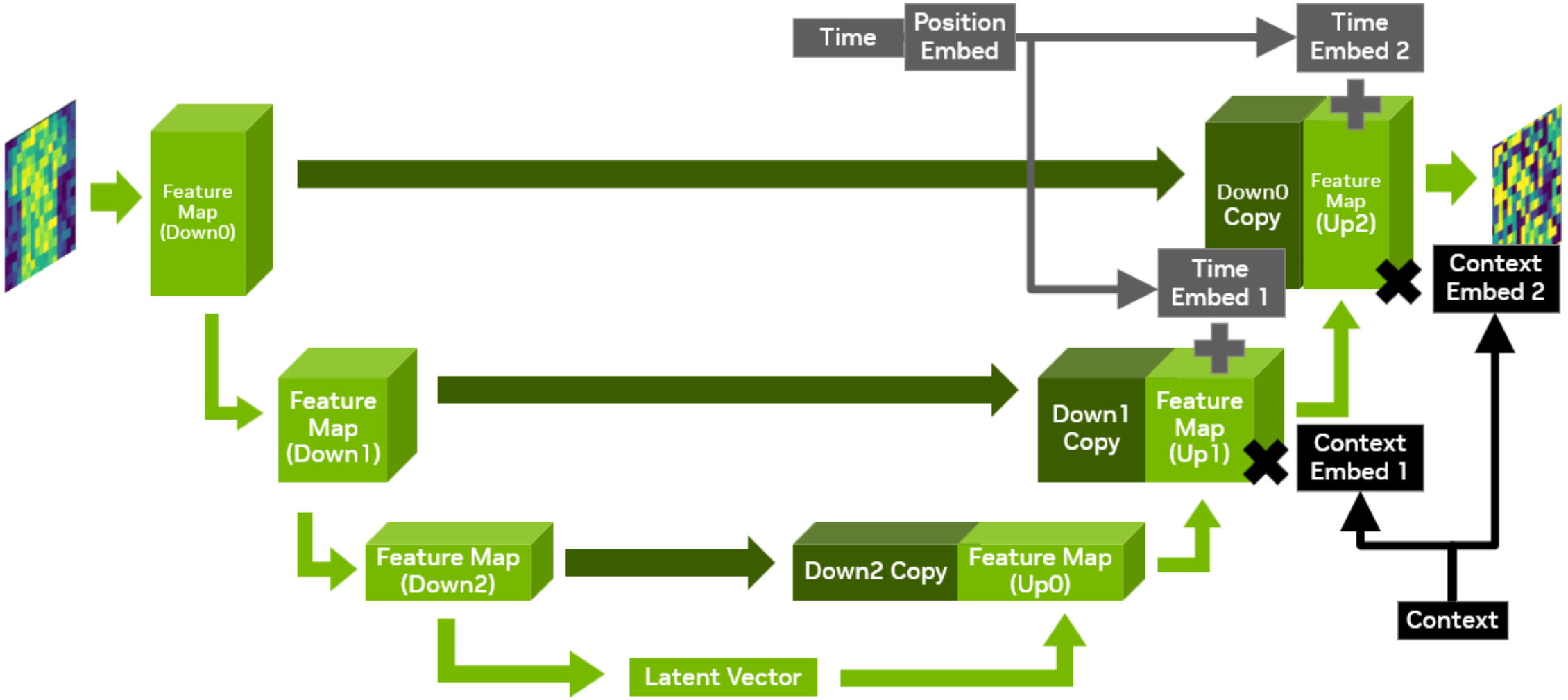
Encoder

Decoder

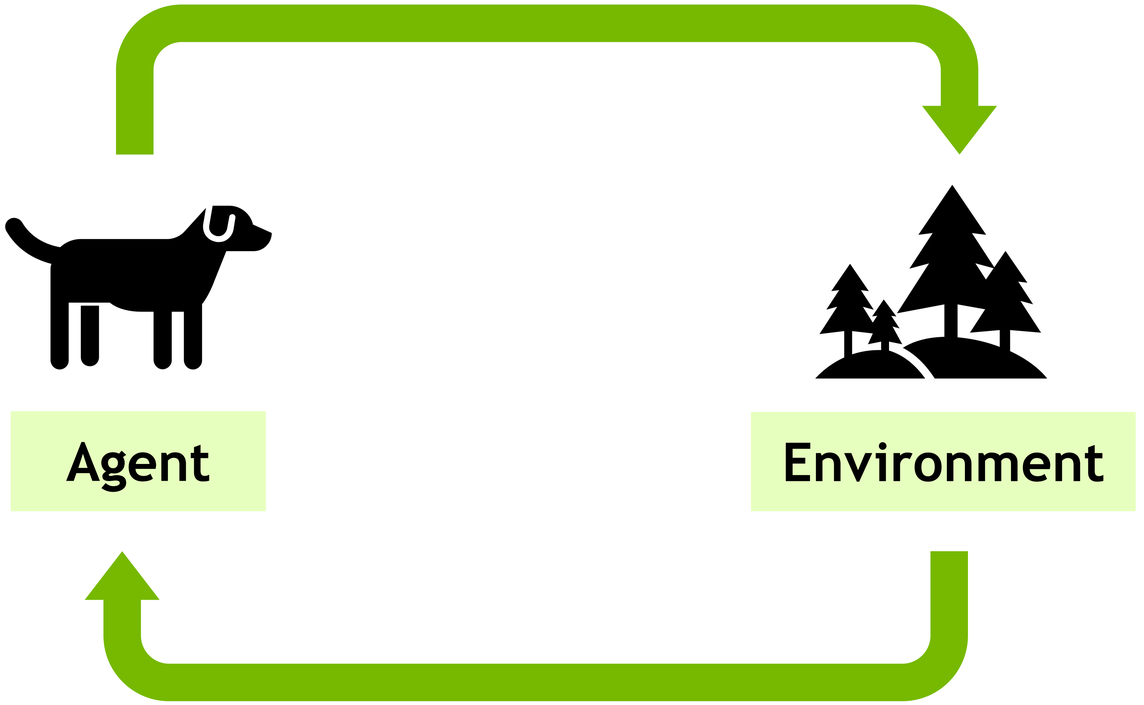
Variational Autoencoder



Diffusion Models



Reinforcement Learning



Next Steps

ENABLING PORTABILITY WITH NGC CONTAINERS

Extensive

- Diverse range of workloads and industry specific use cases

Optimized

- DL containers updated monthly
- Packed with latest features and superior performance

Secure & Reliable

- Scanned for vulnerabilities and crypto
- Tested on workstations, servers, & cloud instances

Scalable

- Supports multi-GPU & multi-node systems

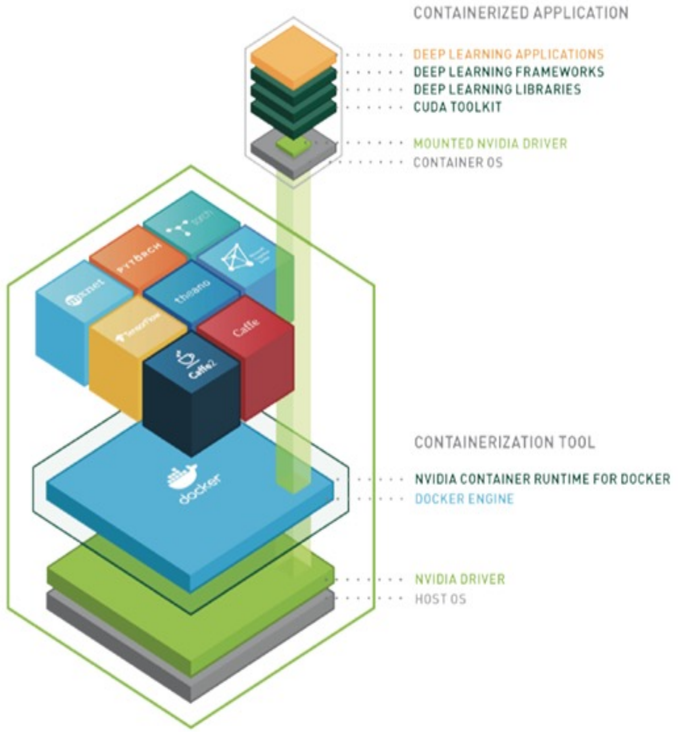
Designed for Enterprise & HPC

- Supports Docker, Singularity & other runtimes

Run Anywhere

- Bare metal, VMs, Kubernetes
- x86, ARM, POWER
- Multi-cloud, on-prem, hybrid, edge

NGC Deep Learning Containers



CONVERSATIONAL AI — Riva

HEALTHCARE — CLARA

SMART CITIES — DEEPSTREAM & SMART PARKING

TELECOM — AERIAL

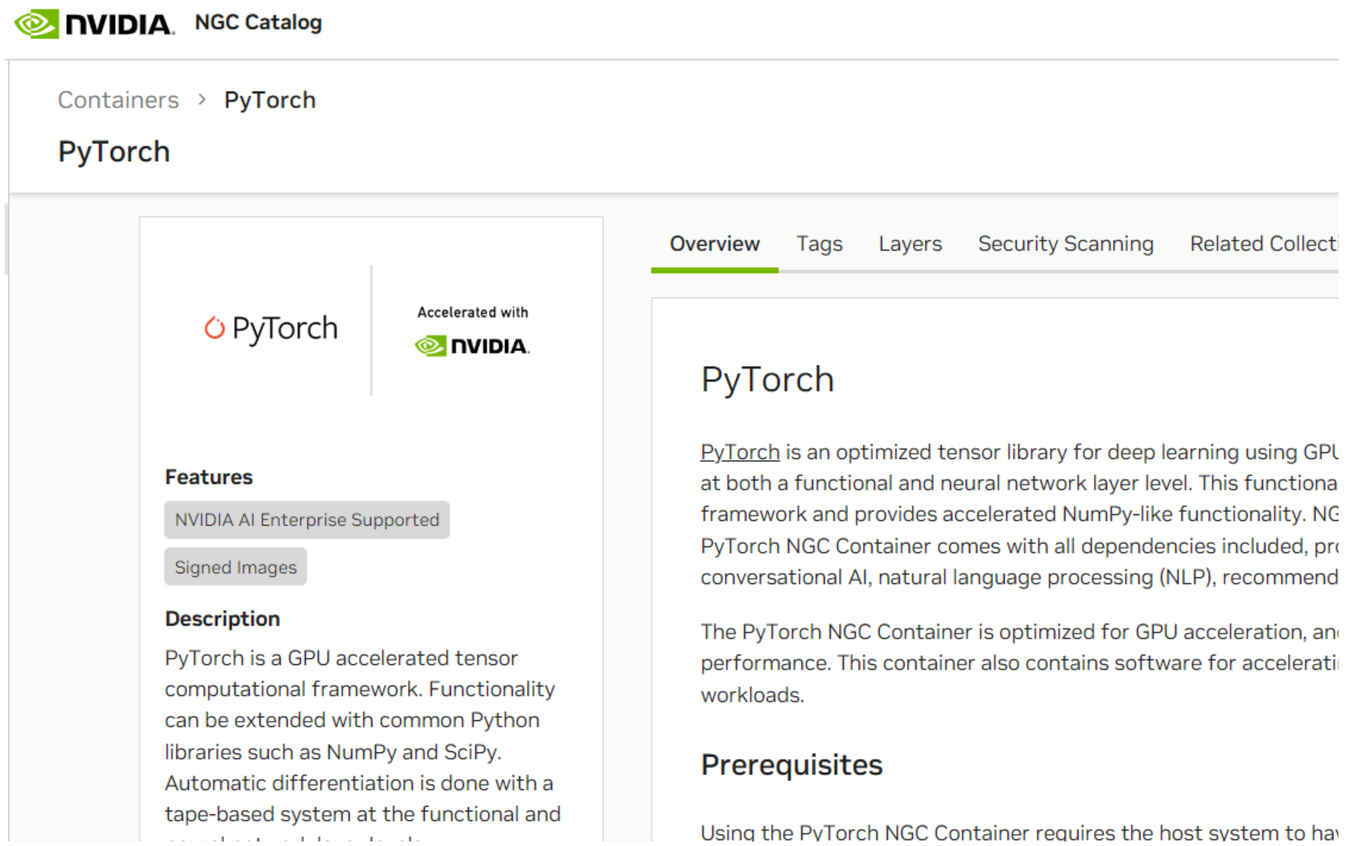
AUTONOMOUS DRIVING — DRIVE

ROBOTICS — ISAAC

HPC — HPC SDK

Learn more about NGC Containers

Next Steps for This Class



Step 1 Sign up for NGC

https://docs.nvidia.com/dgx/ngc-registry-for-dgx-user-guide/index.html

Step 2 Visit NGC Catalog

https://catalog.ngc.nvidia.com/orgs/nvidia/containers/pytorch

Step 3 Pull and Run Container

Visit localhost:8888 to check out a JupyterLab environment

Closing Thoughts

Copying Rocket Science

Let’s get Started!