Deep Learning Fundamentals

An Interactive Introduction to Artificial Neural Networks and TensorFlow

POLL

Where are you?

- The Americas
- Europe / Middle East / Africa
- Asia-Pacific
- Extra-Terrestrial Space

POLL

What are you?

- Developer / Engineer
- Scientist / Analyst / Statistician / Mathematician
- Other

POLL

What's your level of experience with Deep Learning?

- Minimal
- Some theory
- Some theory and code
- Lots of theory and code

POLL

Use of accompanying Jupyter notebooks for interactive demos and in-training exercises?

- Running them on Unix/Linux
- Running them on Mac
- Running them on Windows
- Viewing only, e.g., on GitHub
- Other

Deep Learning Fundamentals

1. The Unreasonable Effectiveness of Deep Learning
2. Deep Learning with Keras
3. Introduction to TensorFlow
4. Deep Learning with TensorFlow

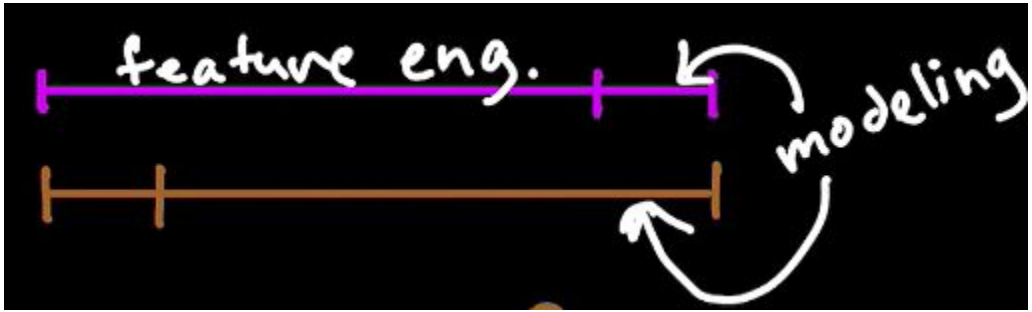
Deep Learning Fundamentals

1. **The Unreasonable Effectiveness of Deep Learning**
   - Intro to Neural Networks and Deep Learning
   - The Deep Learning Families and Libraries
   - Learning with Artificial Neurons
   - TensorFlow Playground
2. Deep Learning with Keras
3. Introduction to TensorFlow
4. Deep Learning with TensorFlow

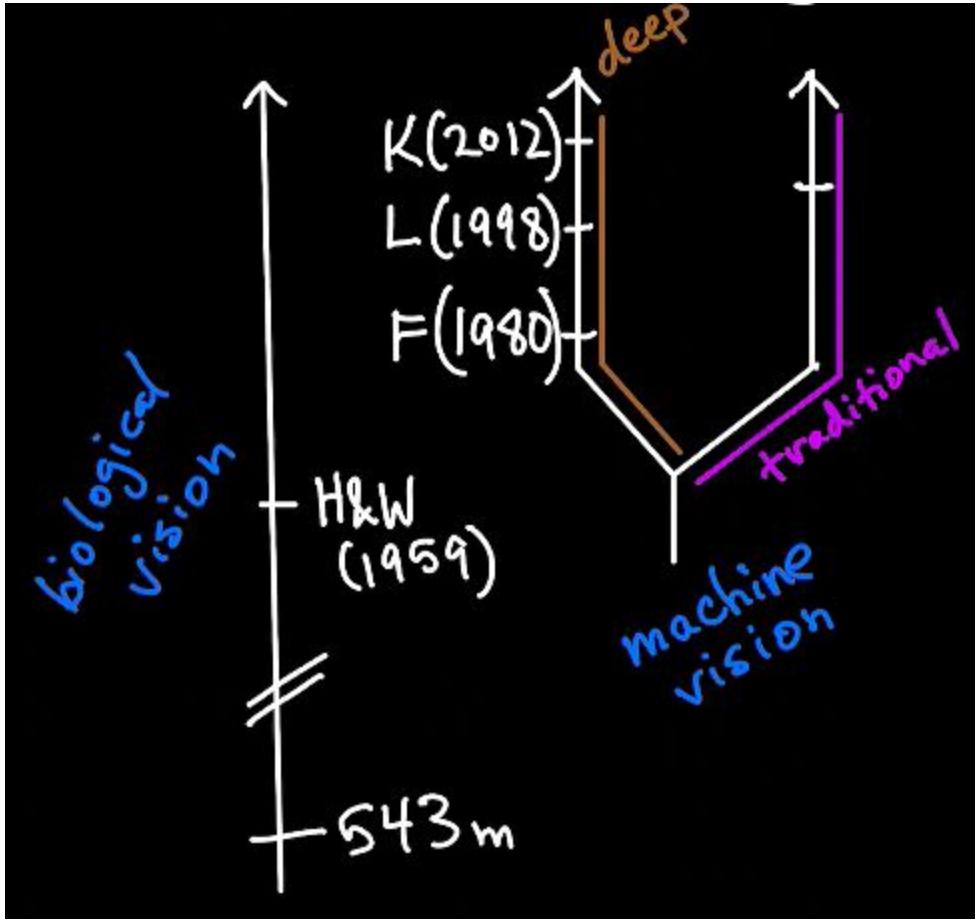
Deep Learning Fundamentals

1. **The Unreasonable Effectiveness of Deep Learning**
   - **Intro to Neural Networks and Deep Learning** *(reference LiveLessons section 1.1)*
   - The Deep Learning Families and Libraries
   - Learning with Artificial Neurons
   - TensorFlow Playground
2. Deep Learning with Keras
3. Introduction to TensorFlow
4. Deep Learning with TensorFlow

Traditional ML vs Deep Learning



Case Study: The History of Vision



Neurocognitron (Fukushima, 1980)

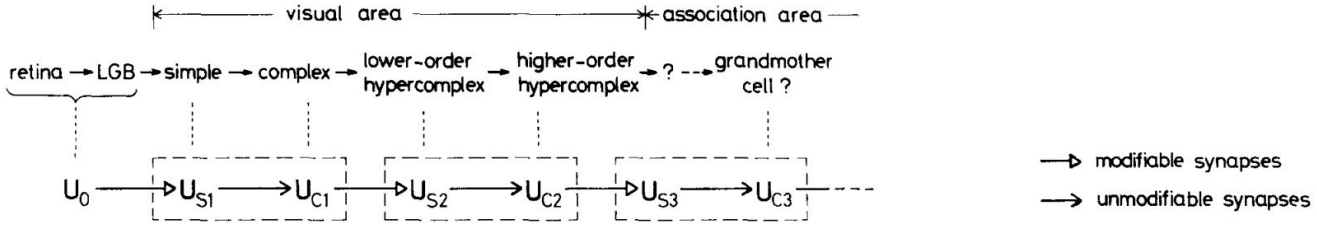


Fig. 1. Correspondence between the hierarchy model by Hubel and Wiesel, and the neural network of the neocognitron

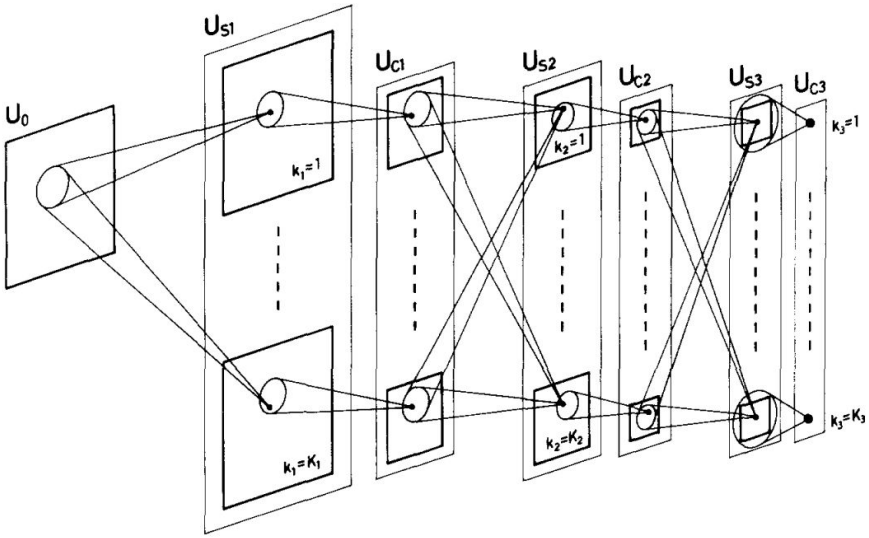
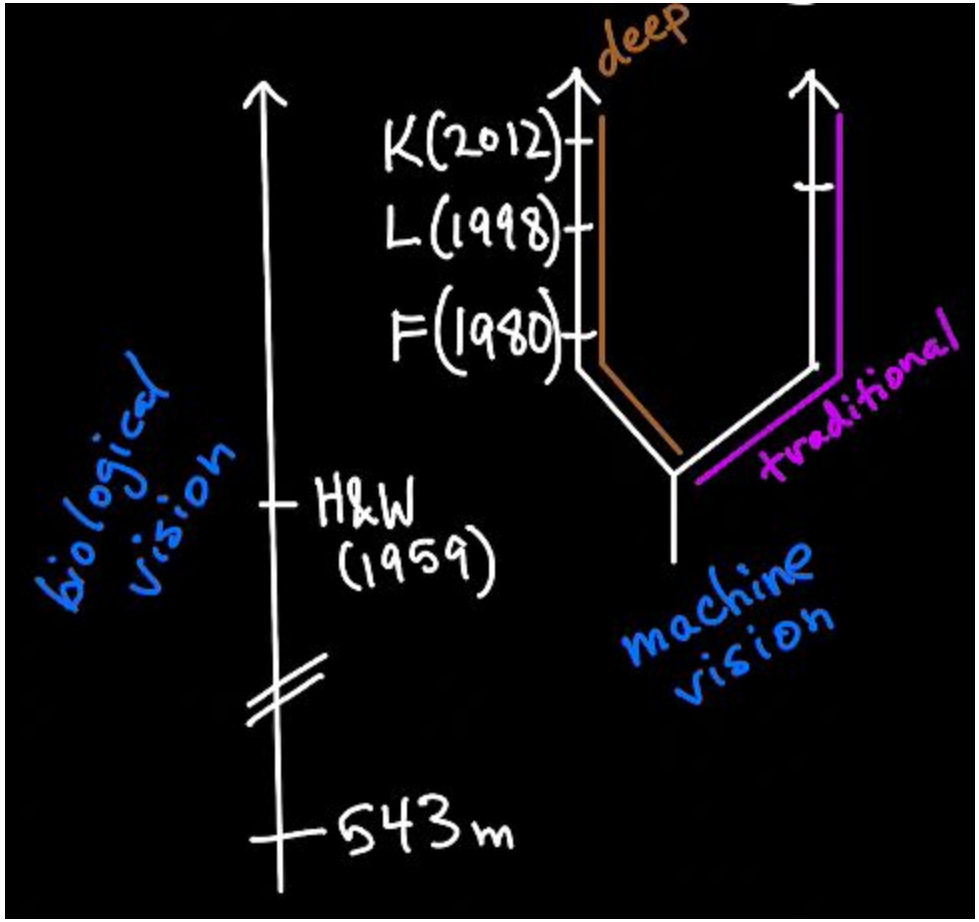
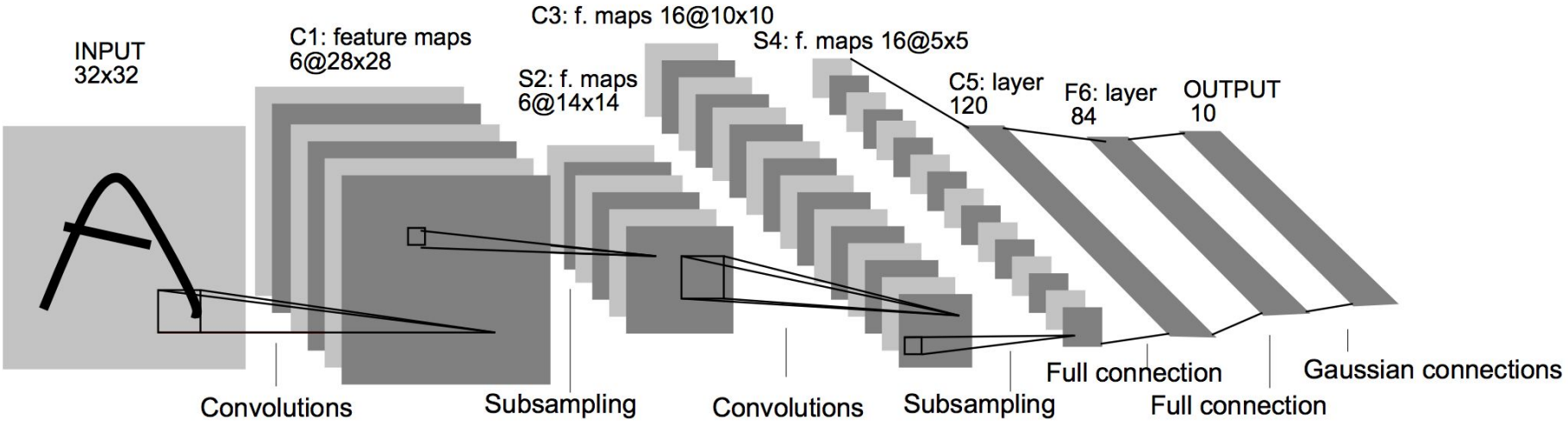
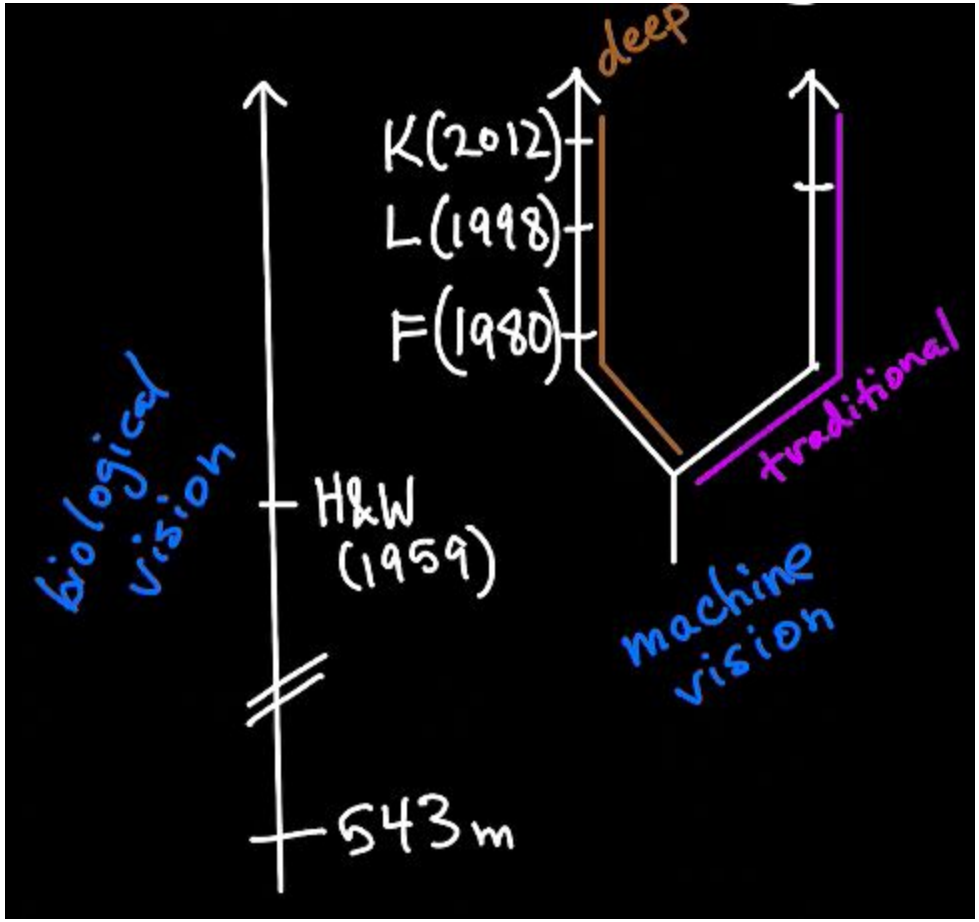


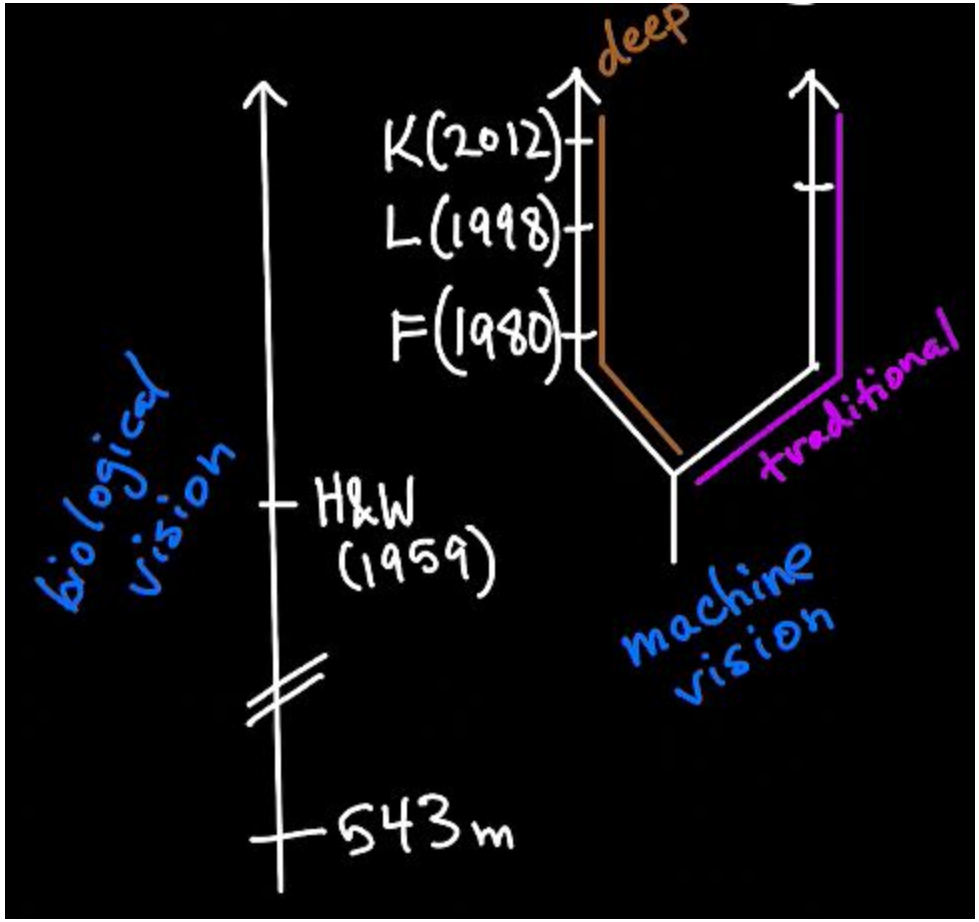
Fig. 2. Schematic diagram illustrating the interconnections between layers in the neocognitron



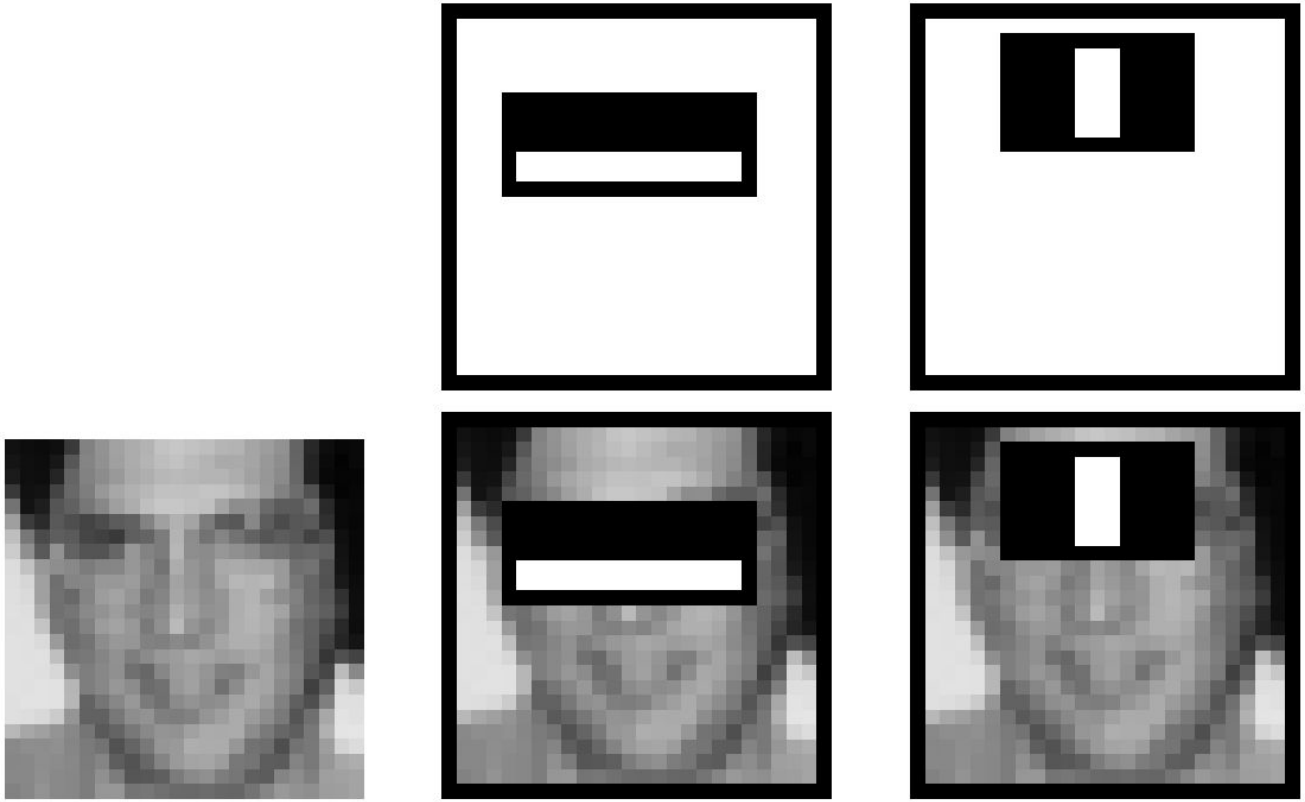
LeNet-5 (LeCun et al., 1998)

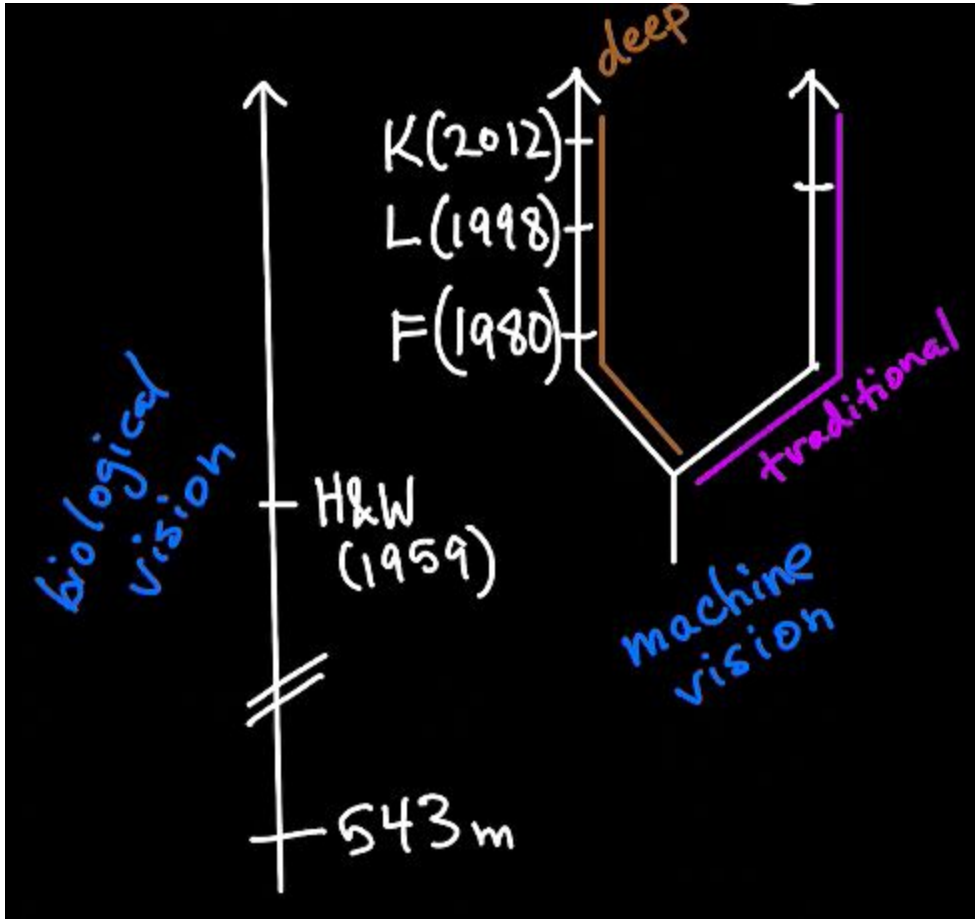




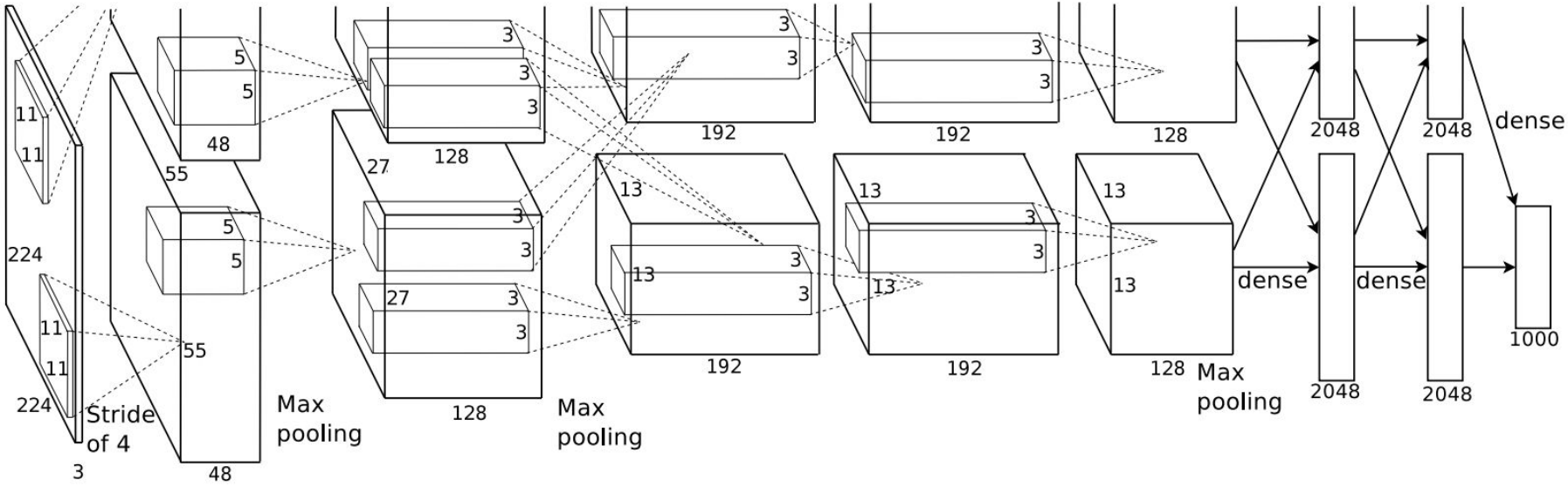


Viola & Jones (2001)





AlexNet (Krizhevsky et al., 2012)



POLL

If a voice recognition algorithm is fed audio of speech as inputs, given corresponding text as the outputs (labels) to learn, and no features are explicitly programmed, is this a:

- Traditional Machine Learning Algorithm
- Deep Learning Algorithm
- I Don't Know

Deep Learning Fundamentals

1. **The Unreasonable Effectiveness of Deep Learning**
   - Intro to Neural Networks and Deep Learning
   - **The Deep Learning Families and Libraries** *(reference LiveLessons sections 2.1 & 4.1)*
   - Learning with Artificial Neurons
   - TensorFlow Playground
2. Deep Learning with Keras
3. Introduction to TensorFlow
4. Deep Learning with TensorFlow

Dense Networks

ConvNets: Convolutional Networks

RNNs: Recurrent Neural Networks

Deep Reinforcement Learning

GANs: Generative Adversarial Networks

POLL

If you were designing an algorithm to learn to play Tetris by maximizing its score, which of these Deep Learning approaches would be most appropriate?

- Convolutional Neural Network
- Recurrent Neural Network
- Deep Reinforcement Learning
- Generative Adversarial Network

POLL

If you were designing an algorithm to recognise tumours in medical images, which of these Deep Learning approaches would be most appropriate?

- Convolutional Neural Network
- Recurrent Neural Network
- Deep Reinforcement Learning
- Generative Adversarial Network

POLL

If you were designing an algorithm to predict stock price movements based on time series data, which of these Deep Learning approaches would be most appropriate?

- Convolutional Neural Network
- Recurrent Neural Network
- Deep Reinforcement Learning
- Generative Adversarial Network

Leading Deep Learning Libraries

	Caffe	Torch	Theano	TensorFlow
language	Python, C++	Lua, PyTorch	Python	Python, Java, C, Go
pre-trained models	Model Zoo	ModelZoo	Lasagne	Inception, others
parallel GPUs: data	Yes	Yes	Yes	Yes
parallel GPUs: model		Yes		Yes
source code	Readable	Readable		
for RNNs			Good	Best
high-level APIs			Keras	Keras, TFLearn

Deep Learning Fundamentals

1. **The Unreasonable Effectiveness of Deep Learning**
   - Intro to Neural Networks and Deep Learning
   - The Deep Learning Families and Libraries
   - **Learning with Artificial Neurons** *(reference LiveLessons sections 2.2, 2.3 & 3.1)*
   - **TensorFlow Playground** *(2.4)*
2. Deep Learning with Keras
3. Introduction to TensorFlow
4. Deep Learning with TensorFlow

Your Arsenal

Neurons

- sigmoid
- tanh
- ReLU

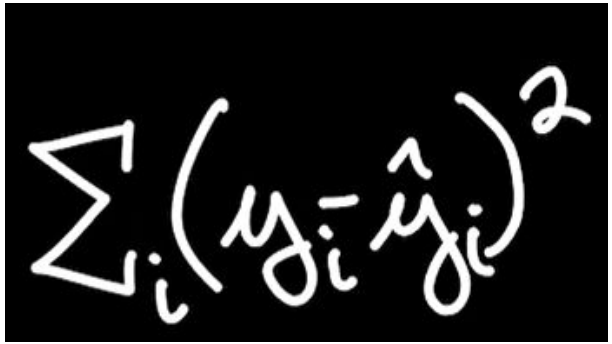
Your Arsenal

Neurons

- sigmoid
- tanh
- ReLU

Cost Functions

- quadratic cost



Your Arsenal

Neurons

- sigmoid
- tanh
- ReLU

Cost Functions

- quadratic cost
- cross-entropy

Your Arsenal

Neurons

- sigmoid
- tanh
- ReLU

Cost Functions

- quadratic cost
- cross-entropy

Gradient Descent

Your Arsenal

Neurons

- sigmoid
- tanh
- ReLU

Cost Functions

- quadratic cost
- cross-entropy

Gradient Descent

Backpropagation

Your Arsenal

Neurons

- sigmoid
- tanh
- ReLU

Cost Functions

- quadratic cost
- cross-entropy

Gradient Descent

Backpropagation

Layers

- dense
- softmax

Your Arsenal

Neurons

- sigmoid
- tanh
- ReLU

Cost Functions

- quadratic cost
- cross-entropy

Gradient Descent

Backpropagation

Layers

- dense
- softmax

Initialization

- Glorot normal
- Glorot uniform

Your Arsenal

Neurons

- sigmoid
- tanh
- ReLU

Cost Functions

- quadratic cost
- cross-entropy

***Stochastic* Gradient Descent**

- mini-batch size
- learning rate
- second-order, e.g., Adam

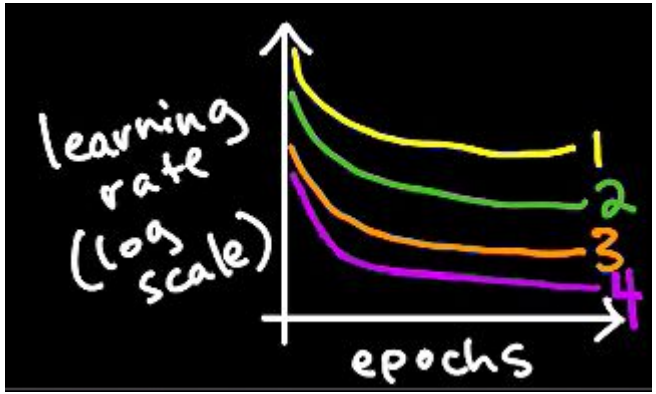
Backpropagation

Initialization

- Glorot normal
- Glorot uniform

Layers

- dense
- softmax



Your Arsenal

Neurons

- sigmoid
- tanh
- ReLU

Cost Functions

- quadratic cost
- cross-entropy

***Stochastic* Gradient Descent**

- mini-batch size
- learning rate
- second-order, e.g., Adam

Backpropagation

Initialization

- Glorot normal
- Glorot uniform

Layers

- dense
- softmax

Avoiding Overfitting

- L1/L2 Regularization
- Dropout
- Data Expansion

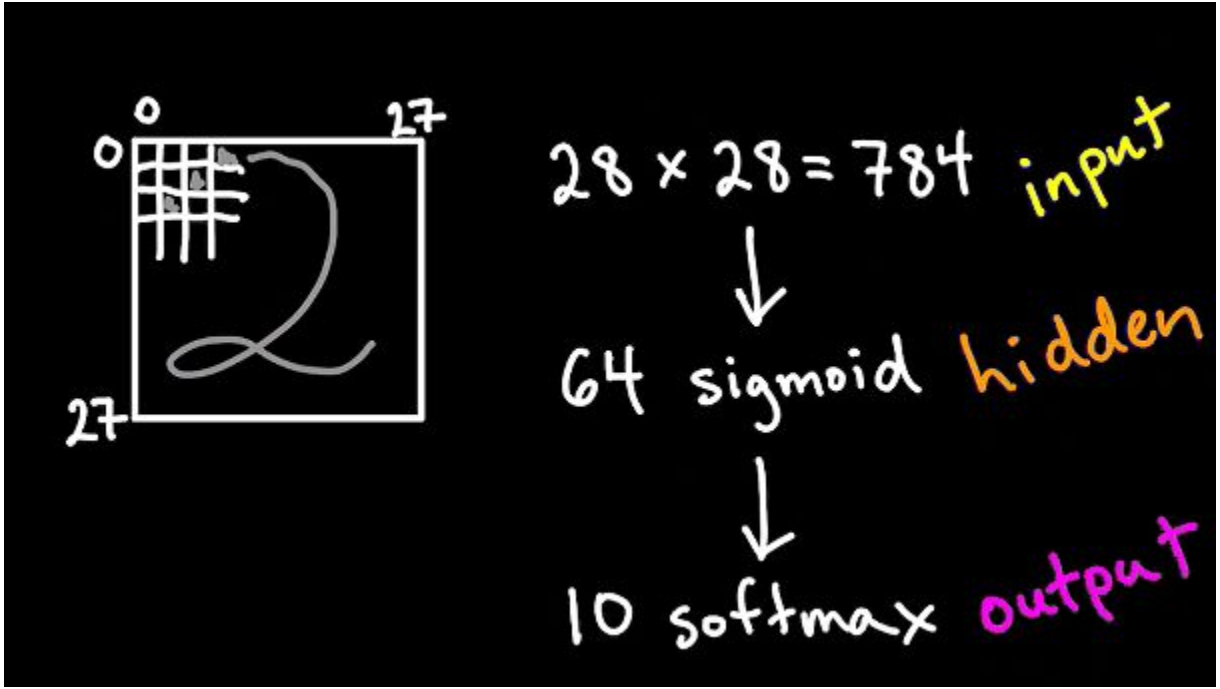
Deep Learning Fundamentals

1. The Unreasonable Effectiveness of Deep Learning
2. **Deep Learning with Keras**
   - **Shallow Neural Networks**
   - **Deep Neural Networks**
   - **How to Build Your Own Project**
3. Introduction to TensorFlow
4. Deep Learning with TensorFlow

Deep Learning Fundamentals

1. The Unreasonable Effectiveness of Deep Learning
2. **Deep Learning with Keras**
   - **Shallow Neural Networks** *(LiveLessons 1.3 & 2.5)*
   - Deep Neural Networks
   - How to Build Your Own Project
3. Introduction to TensorFlow
4. Deep Learning with TensorFlow

MNIST Digits



Deep Learning Fundamentals

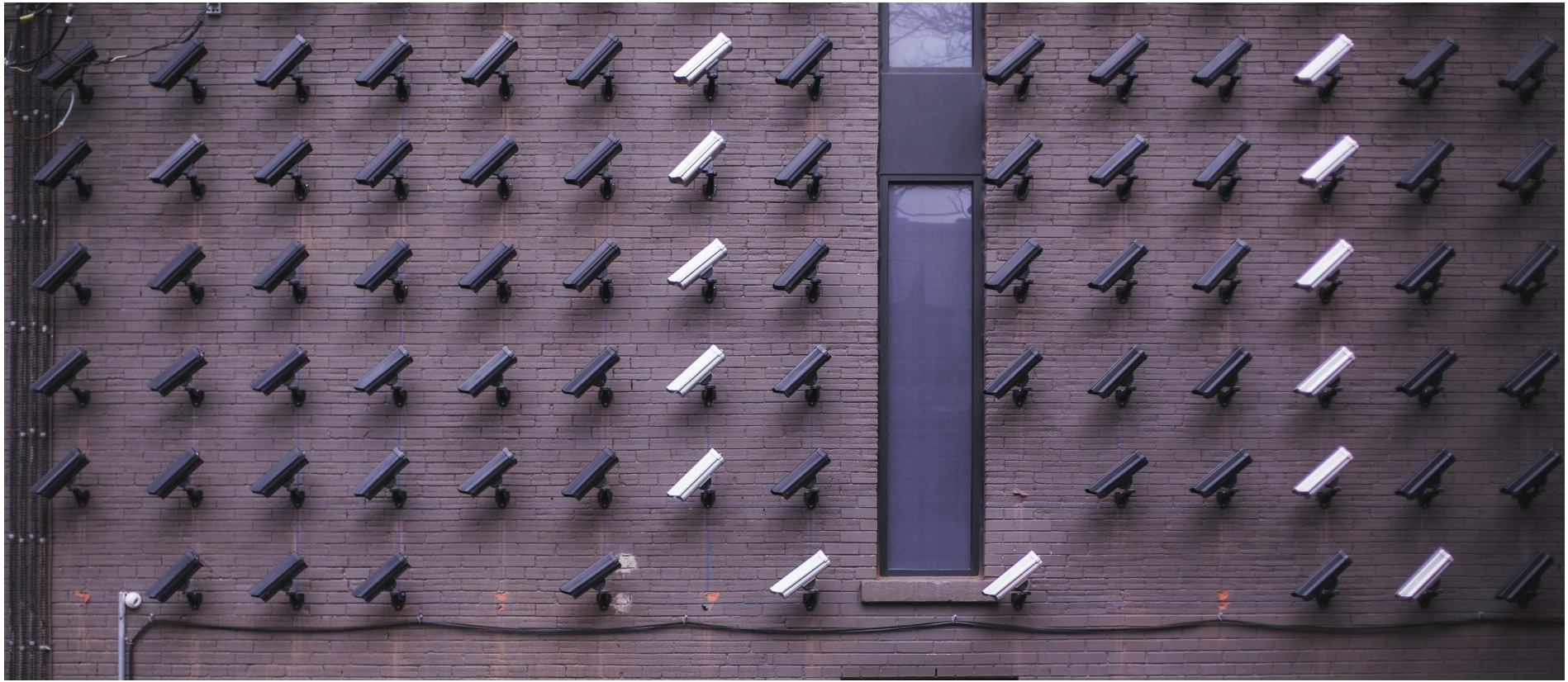
1. The Unreasonable Effectiveness of Deep Learning
2. **Deep Learning with Keras**
   - Shallow Neural Networks
   - Deep Neural Networks *(LiveLessons section 3.2)*
   - How to Build Your Own Project
3. Introduction to TensorFlow
4. Deep Learning with TensorFlow

Deep Learning Fundamentals

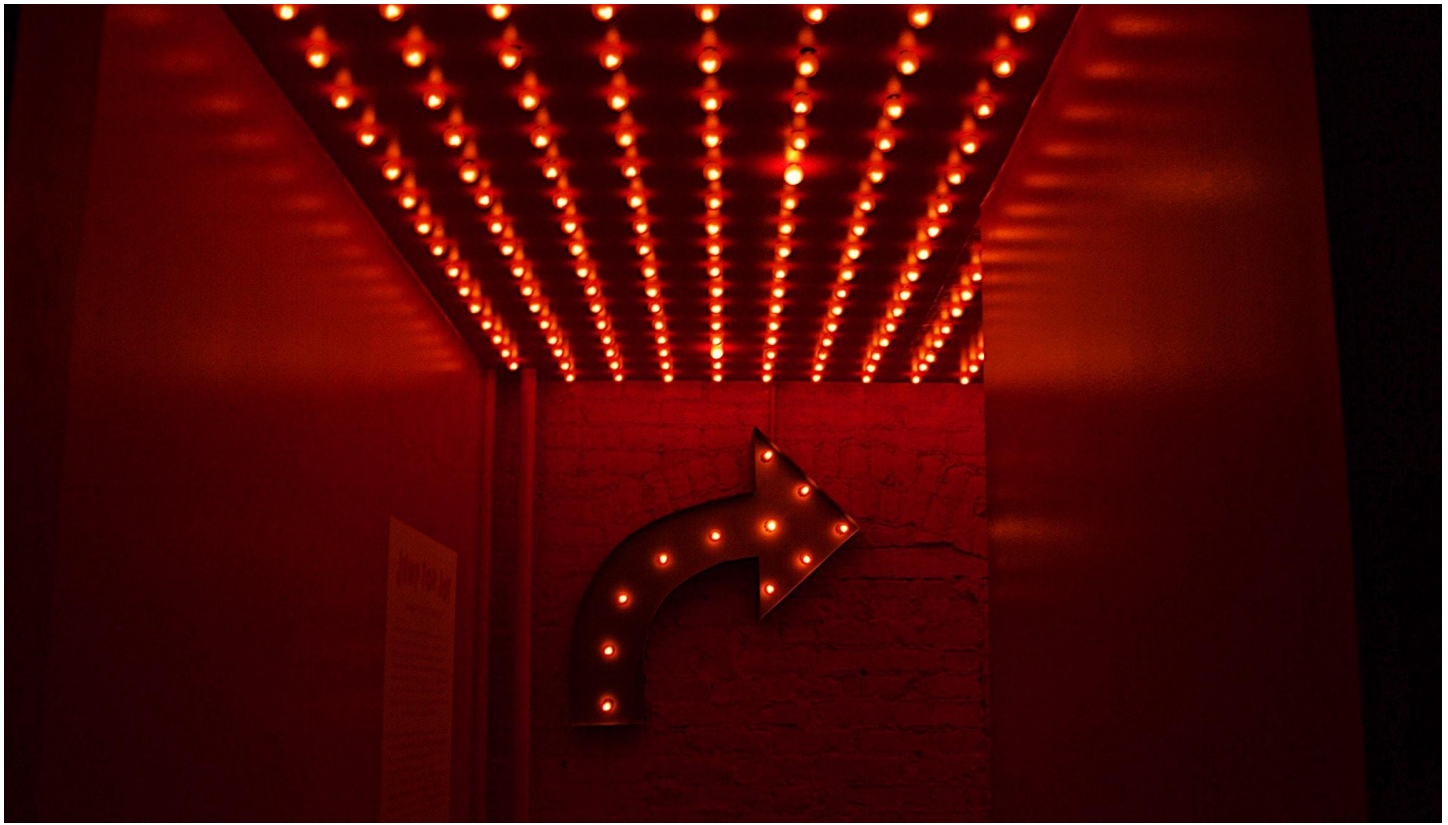
1. The Unreasonable Effectiveness of Deep Learning
2. **Deep Learning with Keras**
   - Shallow Neural Networks
   - Deep Neural Networks
   - How to Build Your Own Project *(LiveLessons 5.1 & 5.2)*
3. Introduction to TensorFlow
4. Deep Learning with TensorFlow

1. Define Task

2. Define Data Set



3. Define Metric

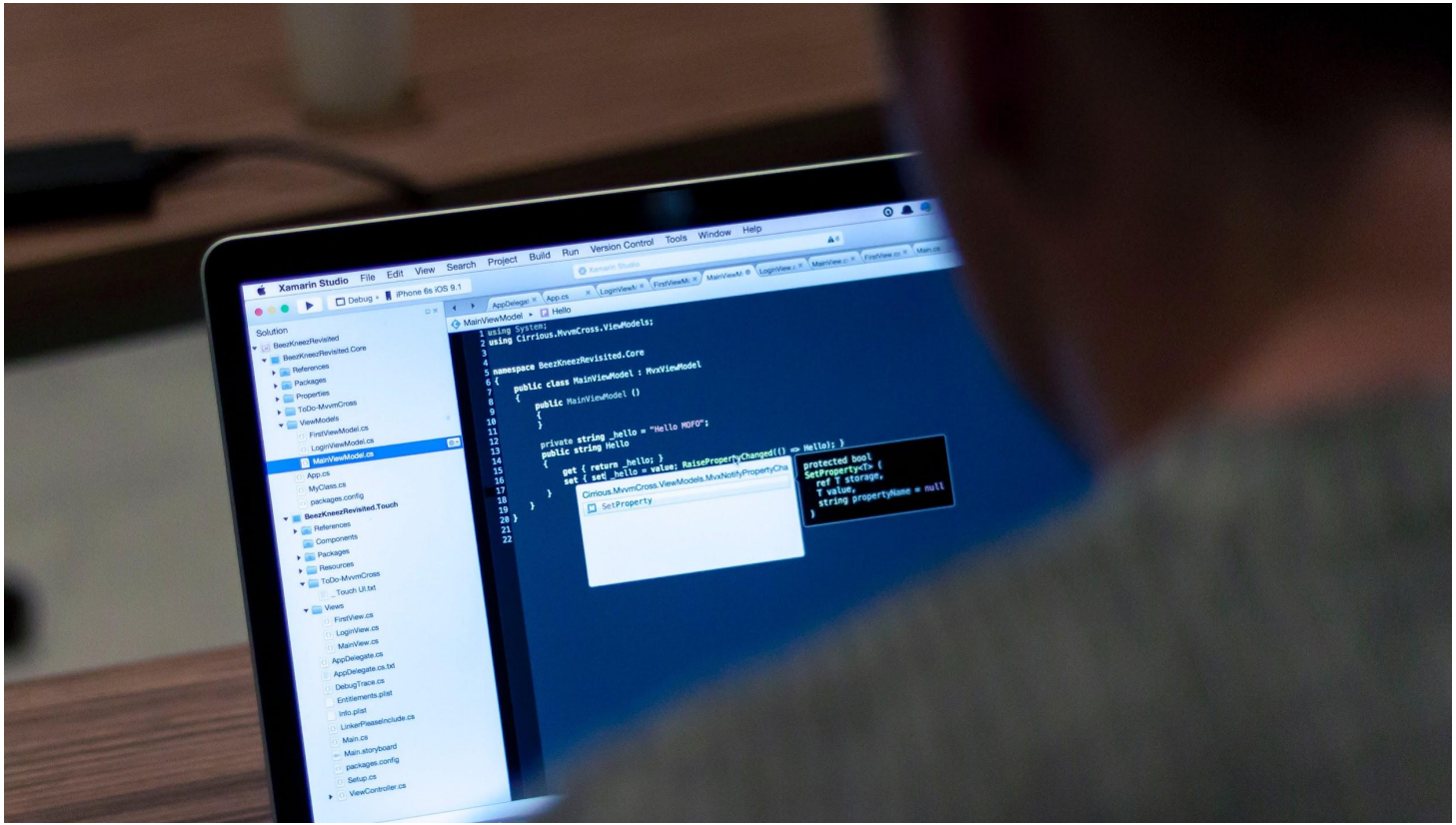


4. Split Data

- Training Set:
  - typically most of your data
  - used for training model weights
- Validation Set:
  - typically a minor subset of your data
  - used for:
    - validating trained model weights
    - tuning model hyperparameters
- Test Set:
  - typically the same size as validation set
  - used rarely to test weights *and* hyperparameters

5. Baseline

6. Implement Existing



7. Improve

...Improving Performance and Tuning Hyperparameters

1. Get Above Chance

If your accuracy is below chance, try:

- simplifying the problem
- simplifying the network architecture
- reducing training set size (to iterate more quickly)

2. Layers

Experiment with varying:

- number of layers
- type of layers
- layer width

3. Initialization

...in `lenet_in_tensorflow.ipynb`:

Set neural network hyperparameters

```
epochs = 20
batch_size = 128
display_progress = 40 # after this many batches, output progress to screen
wt_init = tf.contrib.layers.xavier_initializer() # weight initializer
```

```
weight_dict = {
    'W_c1': tf.get_variable('W_c1',
                            [k_conv_1, k_conv_1, 1, n_conv_1], initializer=wt_init),
    'W_c2': tf.get_variable('W_c2',
                            [k_conv_2, k_conv_2, n_conv_1, n_conv_2], initializer=wt_init),
    'W_d1': tf.get_variable('W_d1',
                            [dense_inputs, n_dense], initializer=wt_init),
    'W_out': tf.get_variable('W_out',
                             [n_dense, n_classes], initializer=wt_init)
}
```

4. Cost

...in `lenet_in_keras.ipynb`:

Configure model

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

...in `lenet_in_tensorflow.ipynb`:

Define model's loss and its optimizer

```
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=predictions, labels=y))
optimizer = tf.train.AdamOptimizer().minimize(cost)
```

5. Avoid Overfitting

If validation cost begins to increase or validation accuracy begins to decrease, consider:

- stopping training earlier
- dropout

5. Avoid Overfitting

...in `lenet_in_keras.ipynb`:

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(n_classes, activation='softmax'))
```

...in `lenet_in_tensorflow.ipynb`:

```
# max pooling layer:
pool_size = 2
mp_layer_dropout = 0.25

# dense layer:
n_dense = 128
dense_layer_dropout = 0.5
```

```
# convolutional and max-pooling layers:
conv_1 = conv2d(square_x, weights['W_c1'], biases['b_c1'])
conv_2 = conv2d(conv_1, weights['W_c2'], biases['b_c2'])
pool_1 = maxpooling2d(conv_2, mp_psize)
pool_1 = tf.nn.dropout(pool_1, 1-mp_dropout)

# dense layer:
flat = tf.reshape(pool_1, [-1, weights['W_d1'].get_shape().as_list()[0]])
dense_1 = dense(flat, weights['W_d1'], biases['b_d1'])
dense_1 = tf.nn.dropout(dense_1, 1-dense_dropout)
```

5. Avoid Overfitting

If validation cost begins to increase or validation accuracy begins to decrease, consider:

- stopping training earlier
- dropout
- expand data set
- regularize

6. Learning Rate

...in `lenet_in_keras.ipynb`:

Configure model

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

...in `lenet_in_tensorflow.ipynb`:

Define model's loss and its optimizer

```
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=predictions, labels=y))
optimizer = tf.train.AdamOptimizer().minimize(cost)
```

7. Epochs

...in `lenet_in_keras.ipynb`:

Train!

```
model.fit(X_train, y_train, batch_size=128, epochs=20, verbose=1, validation_data=(X_test,
y_test))
```

...in `lenet_in_tensorflow.ipynb`:

Set neural network hyperparameters

```
epochs = 20
batch_size = 128
display_progress = 40 # after this many batches, output progress to screen
wt_init = tf.contrib.layers.xavier_initializer() # weight initializer
```

```
with tf.Session() as session:
    session.run(initializer_op)

    print("Training for", epochs, "epochs.")

    # loop over epochs:
    for epoch in range(epochs):
```

8. Regularization ƛ

If using L1 or L2 regularization, consider:

- adjusting ƛ hyperparameter by orders of magnitude

9. Batch Size

...in `lenet_in_keras.ipynb`:

Train!

```
model.fit(X_train, y_train, batch_size=128, epochs=20, verbose=1, validation_data=(X_test,
y_test))
```

...in `lenet_in_tensorflow.ipynb`:

Set neural network hyperparameters

```
epochs = 20
batch_size = 128
display_progress = 40 # after this i
wt_init = tf.contrib.layers.xavier_
```

```
# loop over all batches of the epoch:
n_batches = int(mnist.train.num_examples / batch_size
for i in range(n_batches):

    # to reassure you something's happening!
    if i % display_progress == 0:
        print("Step ", i+1, " of ", n_batches, " in epoch ", epoch+1, ".", sep='')

    batch_x, batch_y = mnist.train.next_batch(batch_size
```

10. Automation

For grid search of hyperparameters, consider:

- sampling values instead of looping over fixed values
- see `github.com/JasperSnoek/spearmint`

Deep Learning Fundamentals

1. The Unreasonable Effectiveness of Deep Learning
2. Deep Learning with Keras
3. **Introduction to TensorFlow**
   - **TensorFlow Graphs**
   - **Neurons in TensorFlow**
4. Deep Learning with TensorFlow

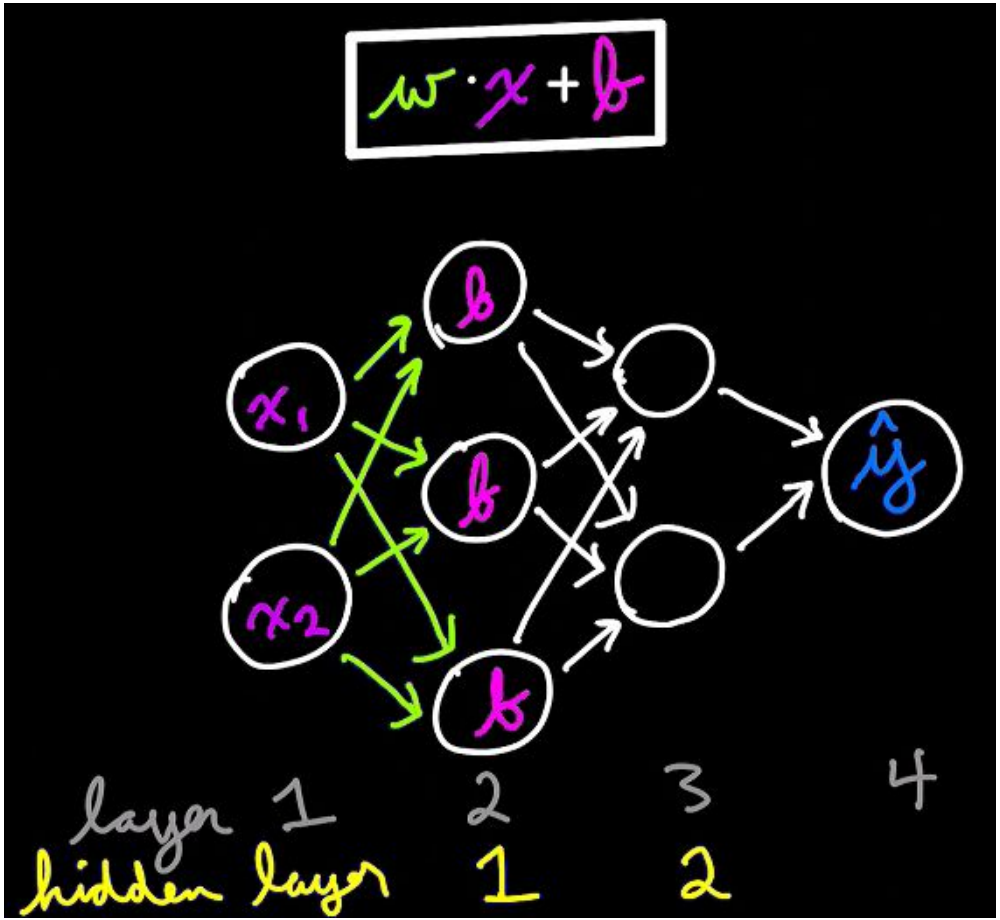
Deep Learning Fundamentals

1. The Unreasonable Effectiveness of Deep Learning
2. Deep Learning with Keras
3. **Introduction to TensorFlow**
   - **TensorFlow Graphs** *(LiveLessons section 4.2)*
   - Neurons in TensorFlow
4. Deep Learning with TensorFlow

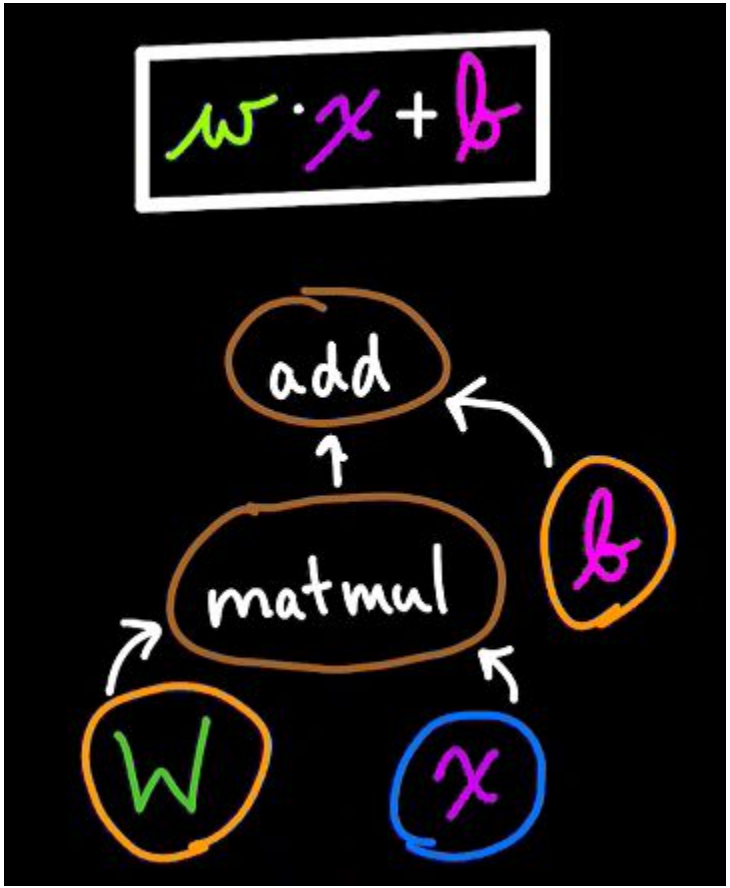
TensorFlow Graphs

1. build graph
2. initialize session
3. fetch and feed data

A Familiar Equation



TensorFlow Graphs



POLL

in a TensorFlow graph, the calculation of a mean is best associated with:

- an operation
- a Variable tensor
- a placeholder tensor

POLL

in a TensorFlow graph, the labels we're predicting with our model are best associated with:

- an operation
- a Variable tensor
- a placeholder tensor

POLL

in a TensorFlow graph, the slope of a regression line is best associated with:

- an operation
- a Variable tensor
- a placeholder tensor

Deep Learning Fundamentals

1. The Unreasonable Effectiveness of Deep Learning
2. Deep Learning with Keras
3. **Introduction to TensorFlow**
   - TensorFlow Graphs
   - **Neurons in TensorFlow** *(LiveLessons section 4.2)*
4. Deep Learning with TensorFlow

Deep Learning Fundamentals

1. The Unreasonable Effectiveness of Deep Learning
2. Deep Learning with Keras
3. Introduction to TensorFlow
4. **Deep Learning with TensorFlow**
   - Model-Fitting
   - Sampling Batches
   - Deep Neural Nets

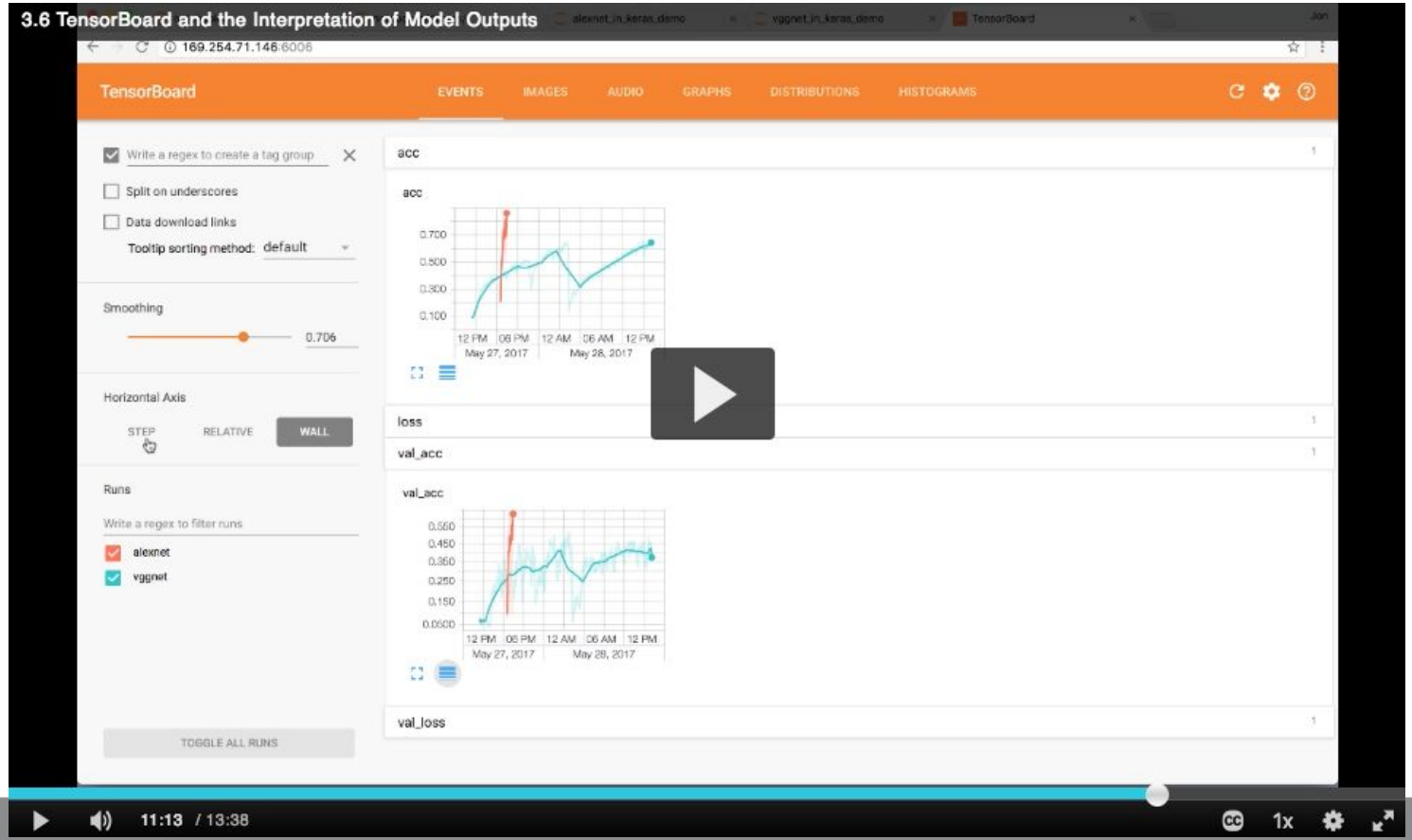
Deep Learning Fundamentals

1. The Unreasonable Effectiveness of Deep Learning
2. Deep Learning with Keras
3. Introduction to TensorFlow
4. **Deep Learning with TensorFlow**
   - **Model-Fitting** *(LiveLessons 4.3)*
   - Sampling Batches
   - Deep Neural Nets

Deep Learning Fundamentals

1. The Unreasonable Effectiveness of Deep Learning
2. Deep Learning with Keras
3. Introduction to TensorFlow
4. **Deep Learning with TensorFlow**
   - Model-Fitting
   - Sampling Batches
   - **Deep Neural Nets** *(LiveLessons 4.4)*

TensorBoard



POLL

what's most useful for follow-up?

- CNNs and Machine Vision
- Sequences: RNNs, LSTMs, NLP, Financial Time Series
- Generative Adversarial Networks
- Reinforcement Learning
- TensorBoard
- more detail on fundamental theory
- Something Else