LECTURE UNITS
MAT391
Topics in Machine Learning

Kwai Wong
University of Tennessee, Knoxville

November 4, 2022

Acknowledgements:

- Support from NSF, UTK, JICS, ICL, NICS
- LAPENNA, www.jics.utk.edu/lapenna, NSF award #202409
- www.icl.utk.edu, cfdlab.utk.edu, www.xsede.org, www.jics.utk.edu/recsem-reu,
- MagmaDNN is a project grown out from the RECSEM REU Summer program supported under NSF award #1659502
- Source code: www.bitbucket.org/icl/magmadnn
- www.bitbucket.org/cfdl/opendnnwheel

INNOVATIVE COMPUTING LABORATORY

NSF

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Unit 4: DNN Computing Ecosystem

- DNN Training Cycle
- Matrix Matrix Multiplication, Performance
- Typical DNN Model
- Object Detection and Segmentation
- GPU, Jetson Nano, Linux Operating system

Supervised Learning

Supervised Learning Data:
(x, y) x is data, y is label

Goal:
Learn a function to map x -> y

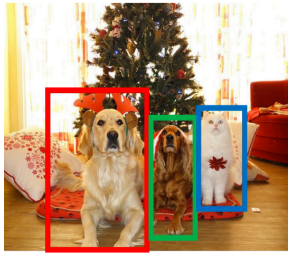
Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

CAT

Classification

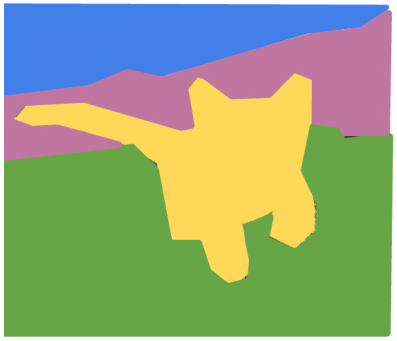
A cat sitting on a suitcase on the floor

Image captioning



DOG, DOG, CAT

Object Detection



GRASS, CAT, TREE, SKY

Semantic Segmentation

Basic Ideas
Typical Neural Network – MLP, CNN

STEP 1 : Model Definition

(Simple Network)

STEP 2 : Cost Function

(Y)

Step 3 : Optimization Scheme

new W = old W– learning rate * (grad [Y w.r.t W])

Step 4 : Numerical Implementation

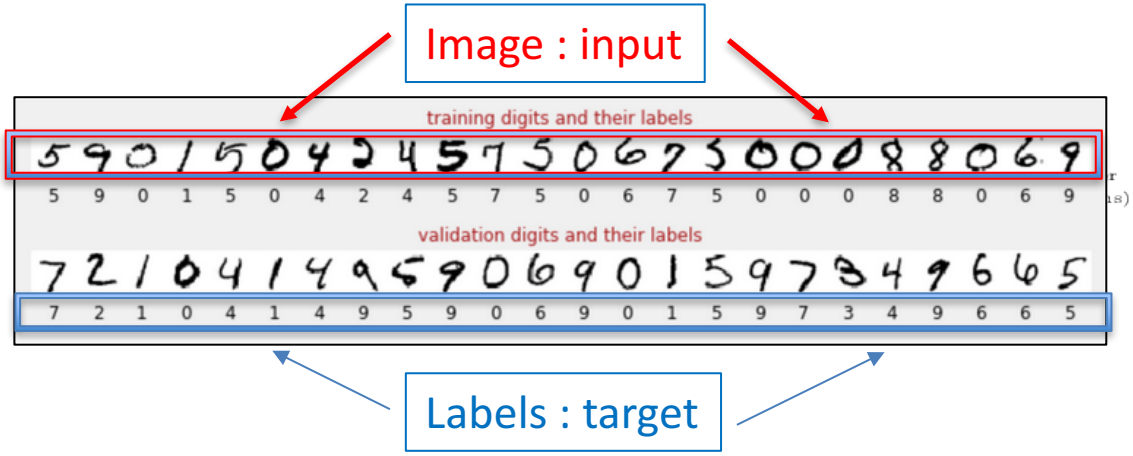
(forward and backward Calculation)

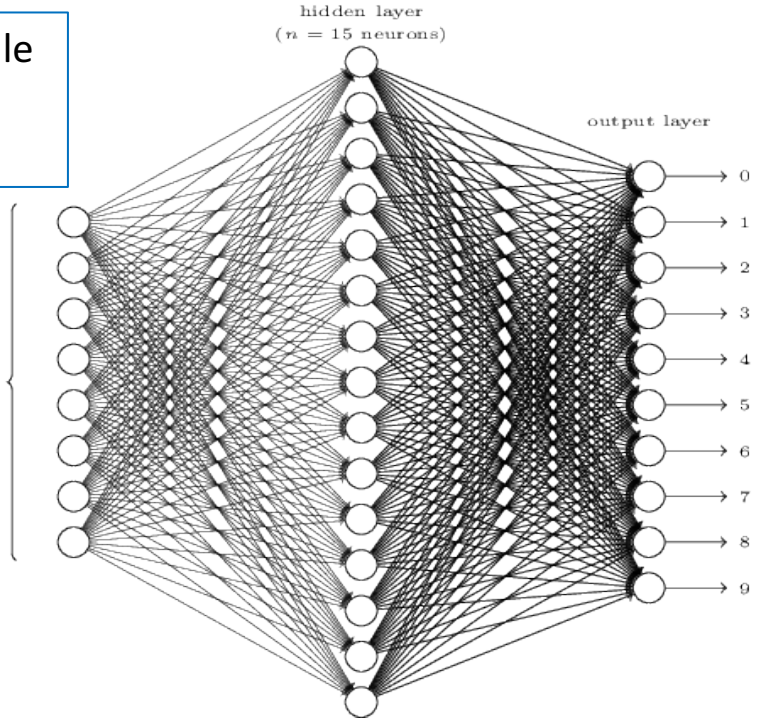
Step 5 : Evaluation

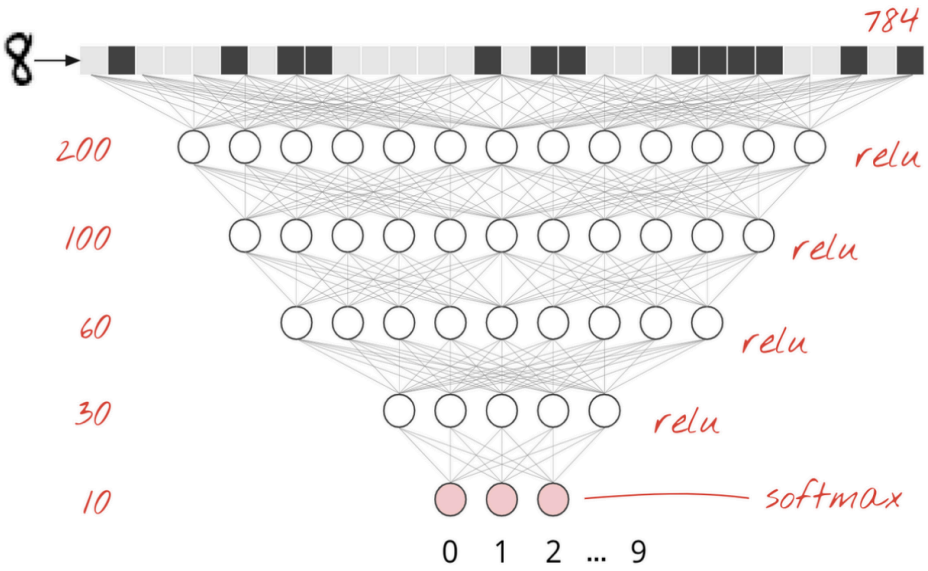
Error (Label – Computed)

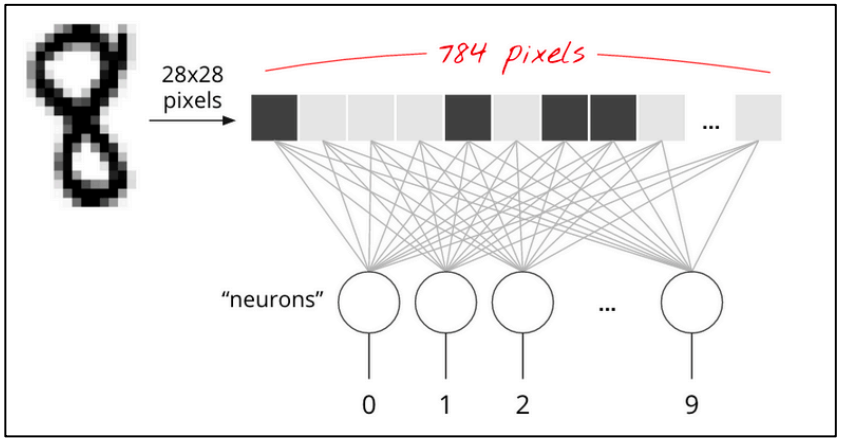
MNIST Example (28x28 pixels)

Handwritten digits in the MNIST dataset are 28x28 pixel grayscale images. The simplest approach for classifying them is to use the 28x28=784 pixels as inputs for a 1-layer neural network.

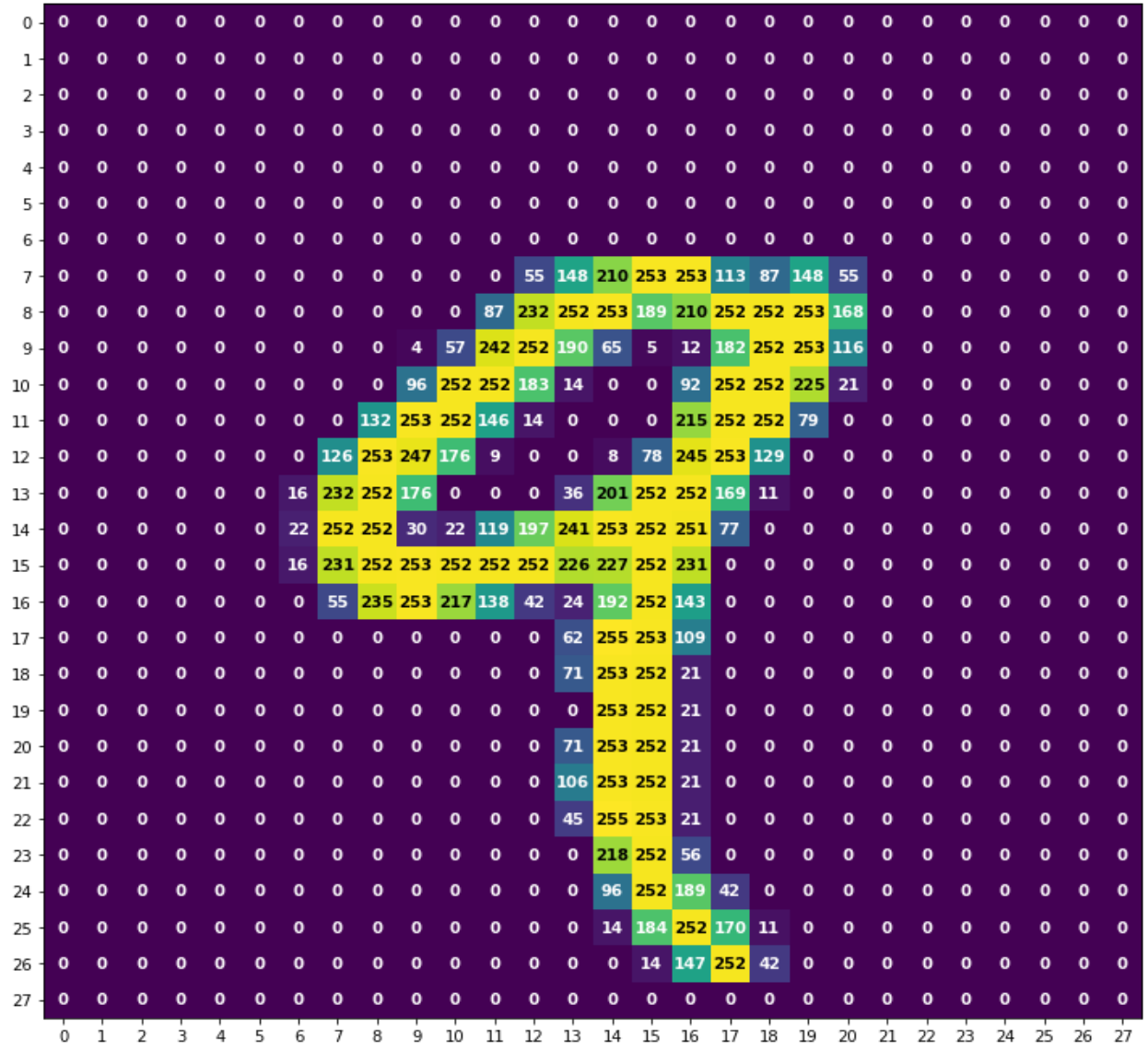


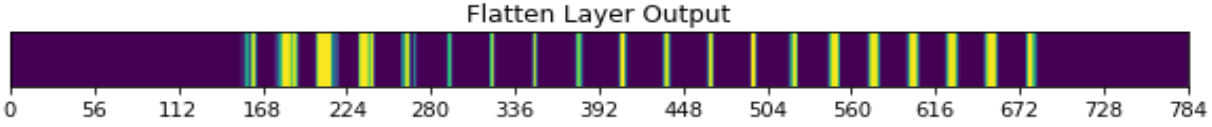






·Flow, Keras and deep learning, without a PhD
https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist#0





Grayscale image
of 28 x 28 pixels

same as a
28 x 28 matrix

values of
0 - 255

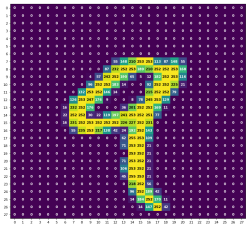
Flatten the

28 x 28 matrix

to a vector of
28 x 28 elements

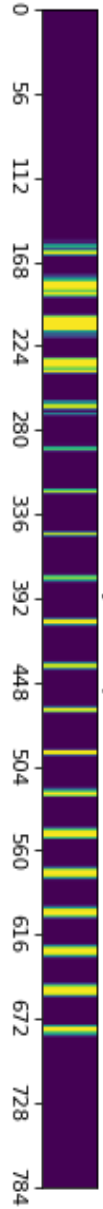
784 elements

Flatten the 28 x 28 matrix to a vector of 28 x 28 of 784 elements vector = Input



Simple MNIST MLP Network

Google Colab Code



Flatten Layer Output

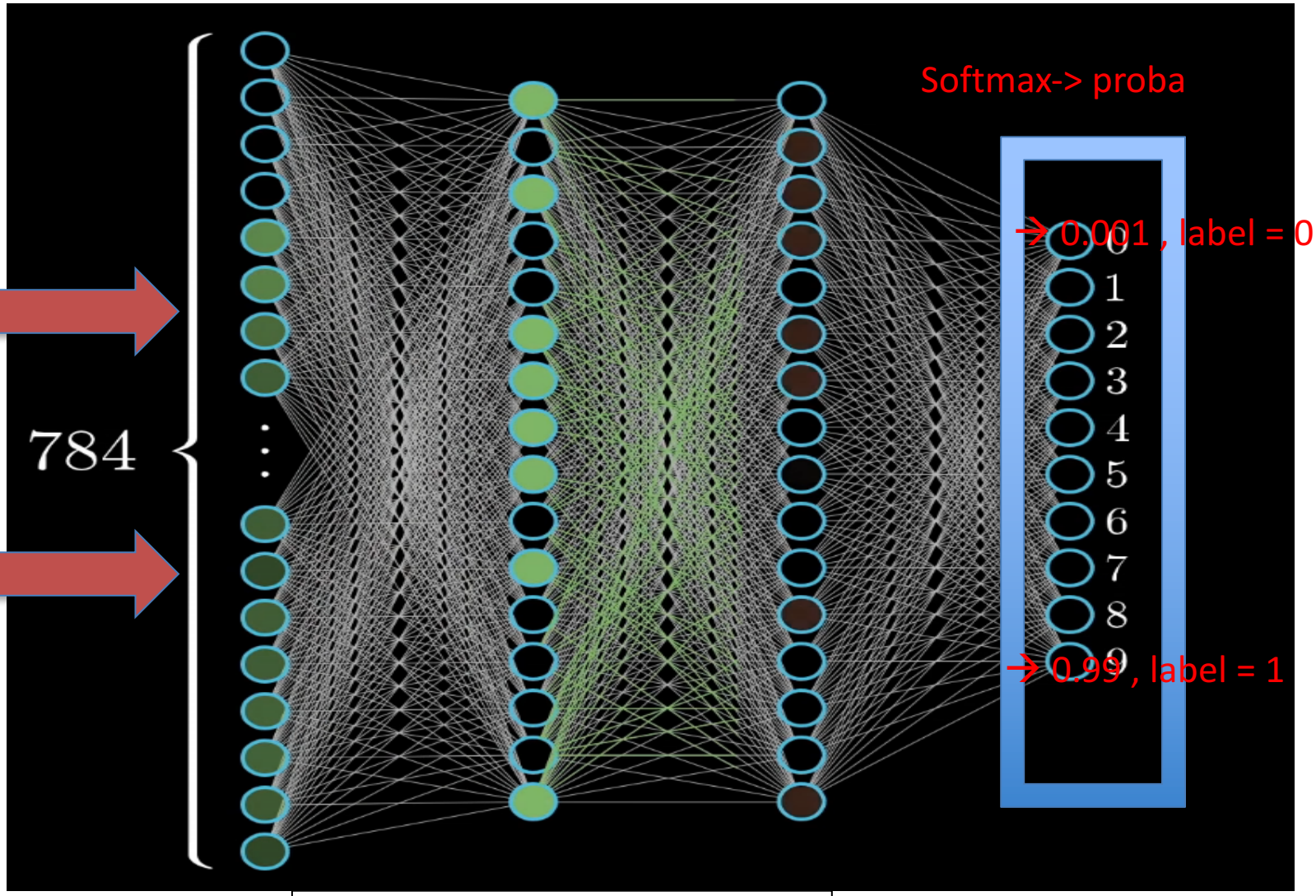
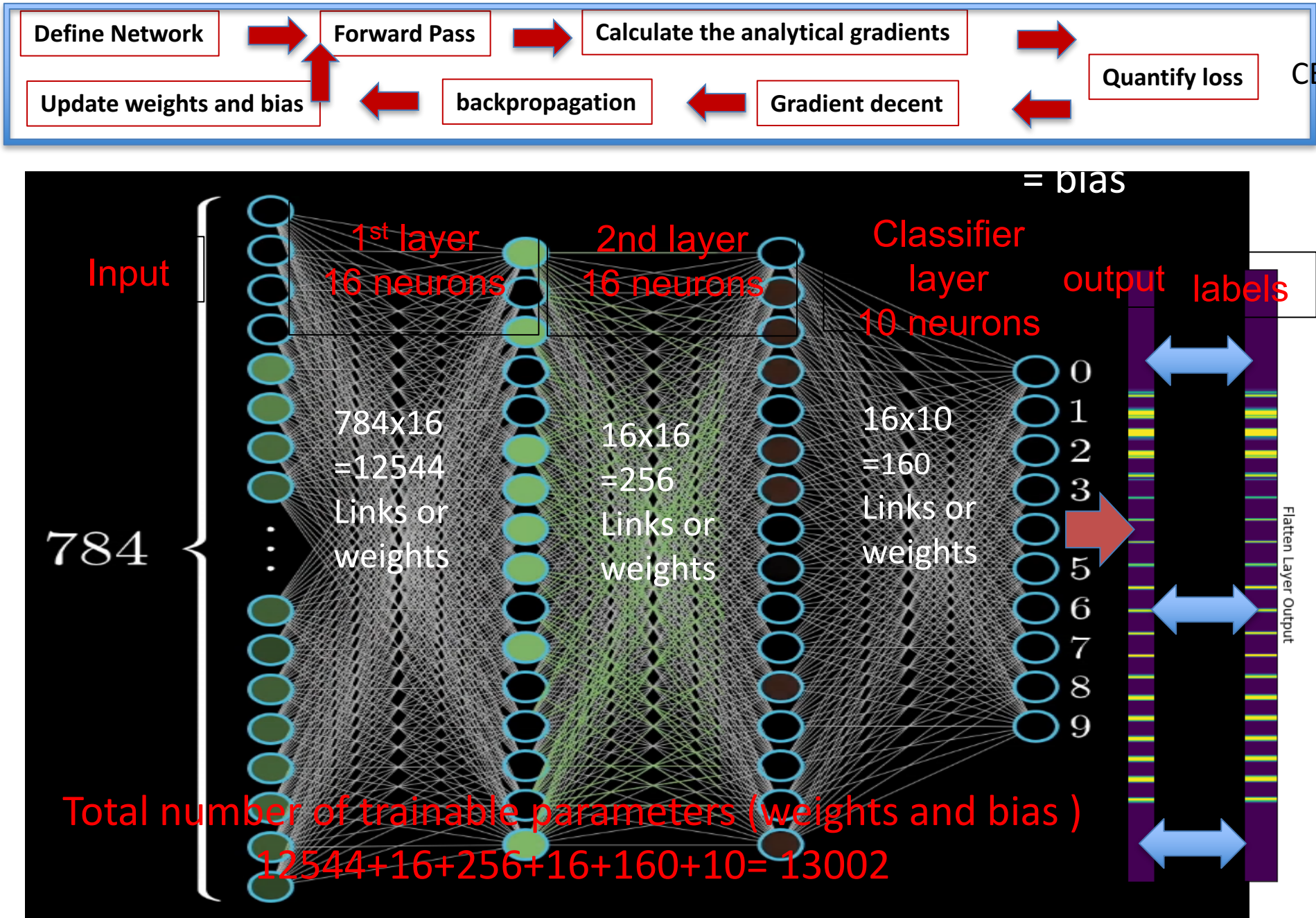


Image:https://www.3blue1brown.com/

```
model.fit(x_train, y_train, epochs=3, batch_size=10)
```



Tensorflow Example

```
import tensorflow as tf

mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(16, activation='relu'),
 tf.keras.layers.Dense(16, activation='relu'),
  tf.keras.layers.Dense(10, activation='softmax')
])
```

Tensorflow is imported

Sets the mnist dataset to variable "mnist"

Loads the mnist dataset

Builds the layers of the model
4 layers in this model

```
model.compile(optimizer='sgd', loss='SparseCategoricalCrossentropy',
metrics=['accuracy'])

model.summary()

logdir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-
%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=1)
```

Loss Function : Y the end of the NN

Compiles the model with the SGD optimizer

Print summary

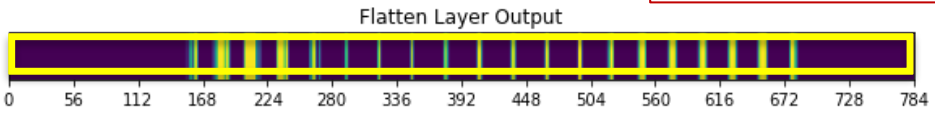
Use tensorborad

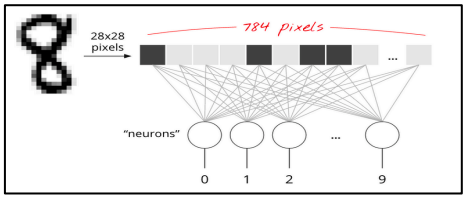
```
model.fit(x_train, y_train, epochs=5, batch_size=10,
validation_data=(x_test, y_test),  callbacks=[tensorboard_callback])

model.evaluate(x_test,  y_test, verbose=2)
%tensorboard --logdir logs
```

Adjusts model parameters to minimize the loss
Tests the model performance on a test set

Summary : Flow of MLP Dense layer NN

1 image = a vector of 784





An image of 28x28 pixels

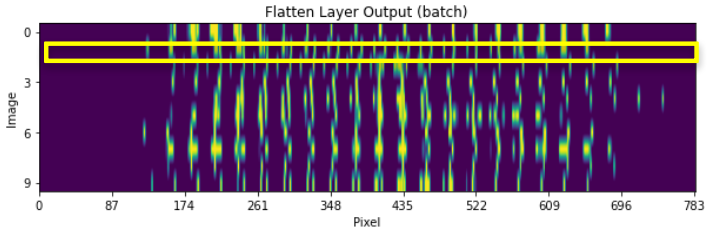
Input to 1st layer

10 images : batch of 10
10 x 784 matrix

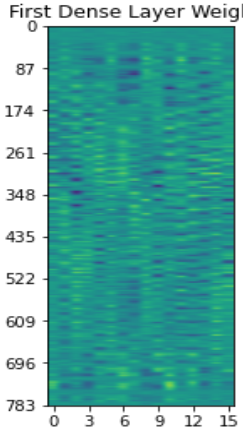
16 neurons
784 x 16 W matrix

10 x 16 output matrix

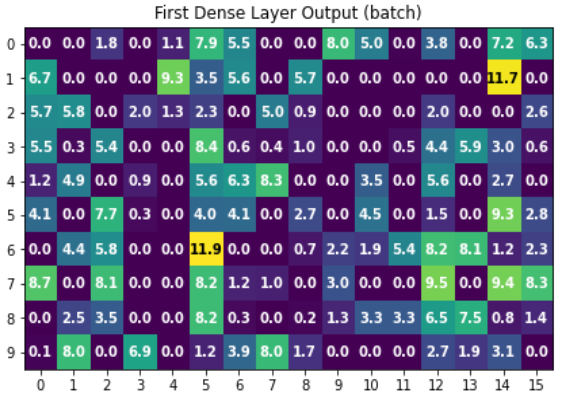
Input to 2nd layer



*

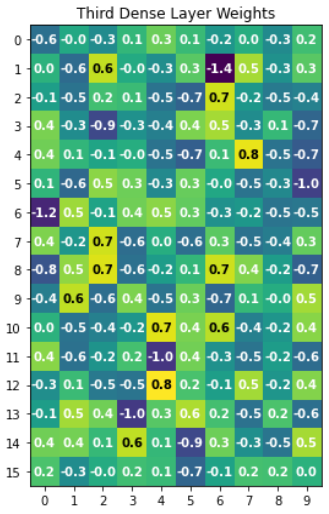


=



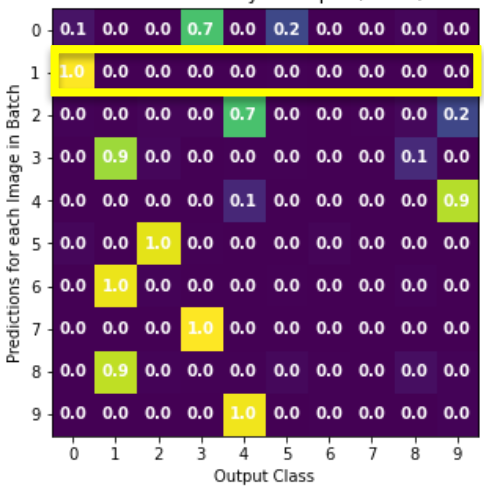
10 x 16 matrix

*

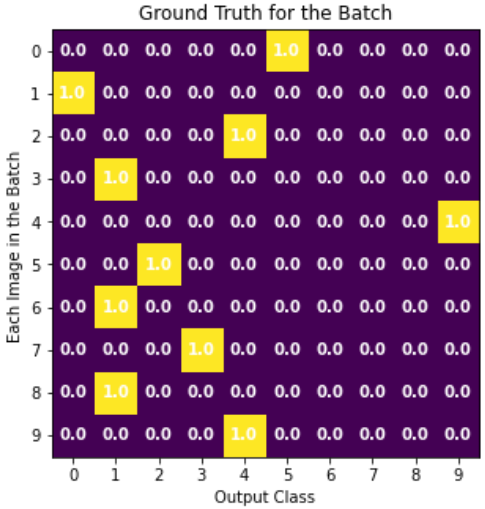


16 neurons
16 x 10
W matrix

=



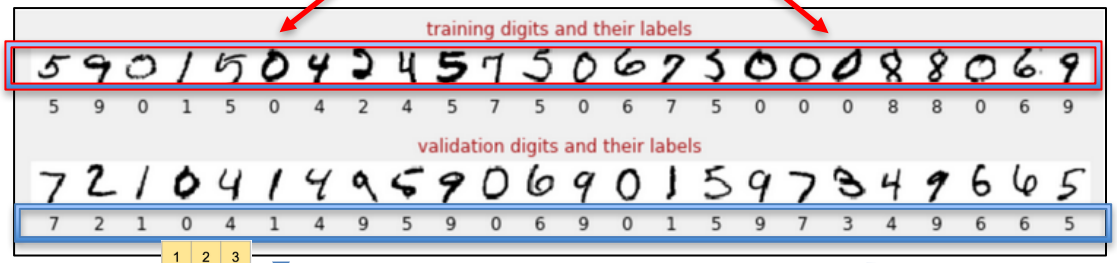
10 x 10 output matrix



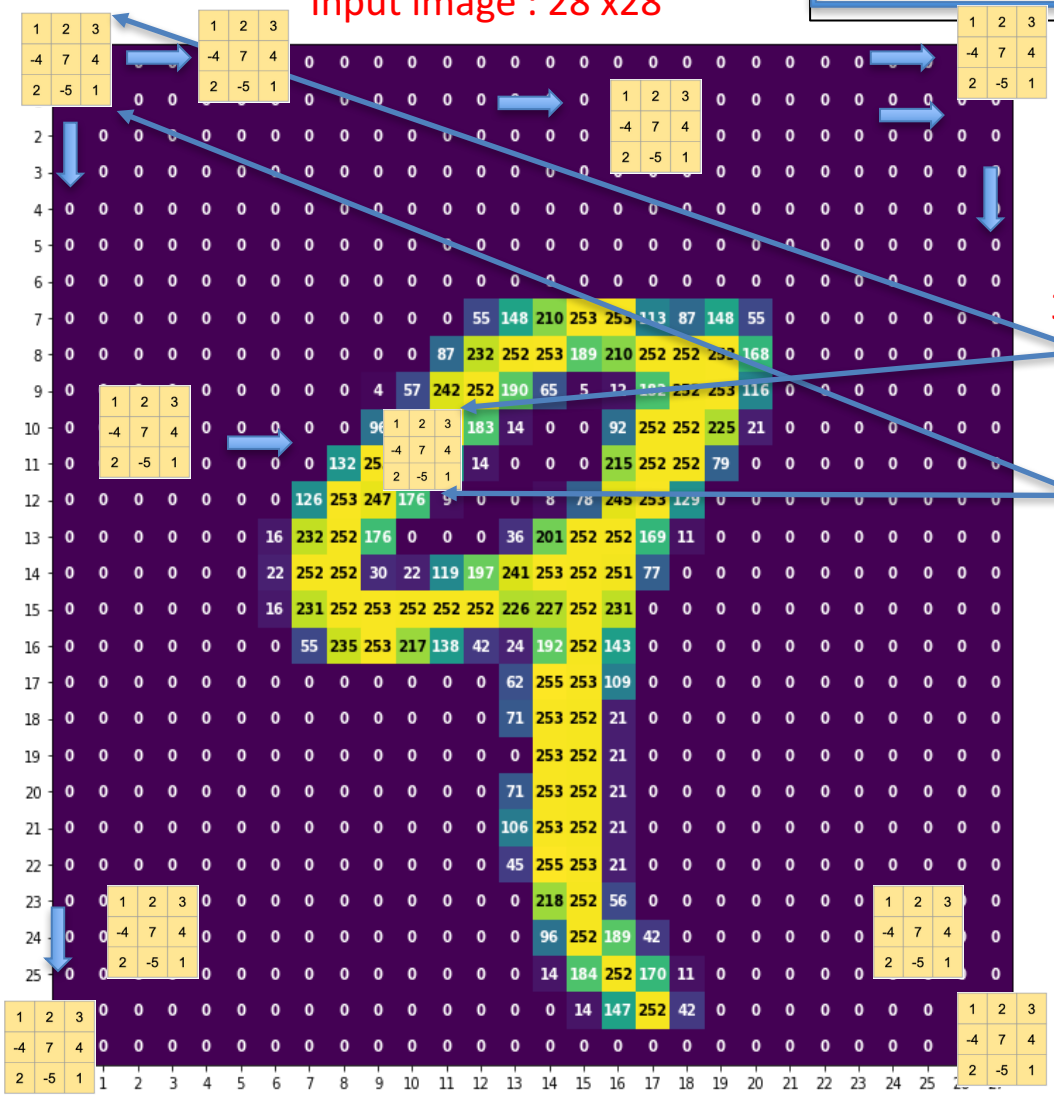
10 x 10 labeled matrix

MNIST Example (28x28 pixels image)

Handwritten digits in the MNIST dataset are 28x28 pixel grayscale images, use the original image (28x28 matrix).

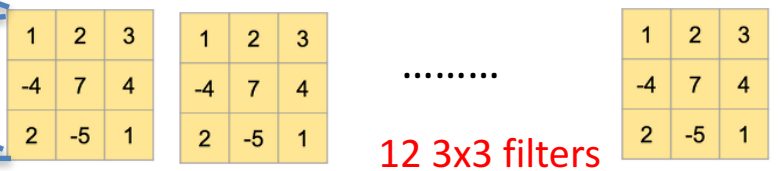


Input image : 28 x28

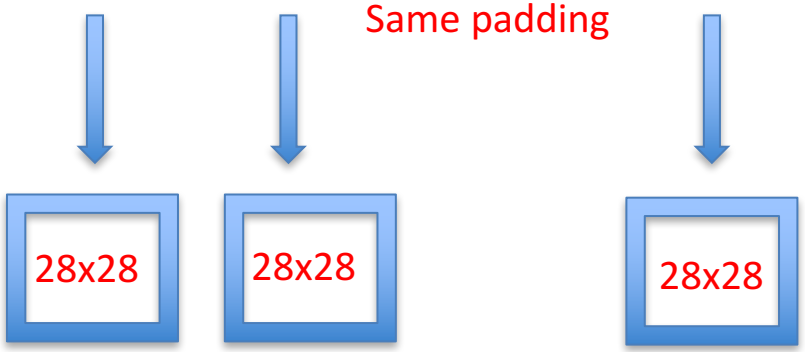


12 filters, each one acts on an image until it is fully covered.

3 x 3 filter (kernel)



12 3x3 filters
Same padding

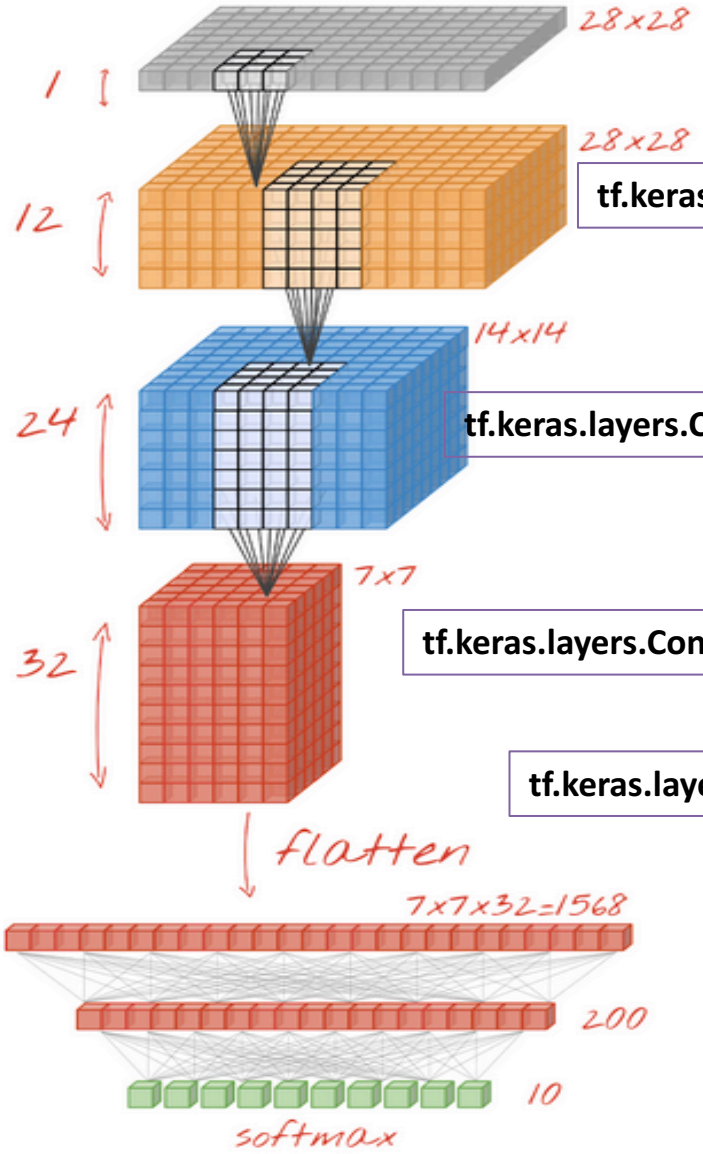


12 Output features : each 28 x28 matrix

Convolution NN : MNIST 28x28 Pixels

[FW, FH, C, H] = [filter size ? X ?, input channel (Depth), output channel (no. of filter)]

tf.keras.layers.Reshape(input_shape=(28*28,), target_shape=(28, 28, 1))



Convolutional 3x3 filters=12

$W_1[3, 3, 1, 12]$

tf.keras.layers.Conv2D(kernel_size=3, filters=12, padding='same', activation='relu')

Convolutional 6x6 filters=24

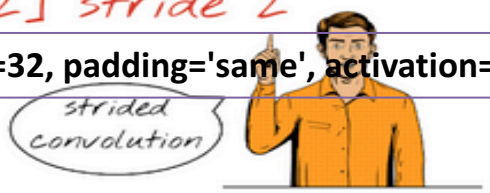
$W_2[6, 6, 12, 24]$ stride 2

tf.keras.layers.Conv2D(kernel_size=6, filters=24, padding='same', activation='relu',strides=2)

Convolutional 6x6 filters=32

$W_2[6, 6, 24, 32]$ stride 2

tf.keras.layers.Conv2D(kernel_size=6, filters=32, padding='same', activation='relu', strides=2)



tf.keras.layers.Flatten()

https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#0

Dense layer

$W_4[1568, 200]$

tf.keras.layers.Dense(200, activation='relu')

Softmax dense layer

$W_5[200, 10]$

tf.keras.layers.Dense(10, activation='softmax')

CNN : example – MNIST

There are three main types of layers in a CNN: Convolutional Layer, Pooling Layer, and Fully-Connected Layer.

- ✓ INPUT [28x28x1] will hold the raw pixel values of the image, in this case an image of width 28, height 28 in grey scale (0-255)
- ✓ Apply twelve (12) 3x3 filters to one layer (channel) - W[3,3,Channel=1,number of filters=12]
- ✓ CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume of [28x28x12] when we use 12 filters. Volume size = [28 width x height = 28 , 12 channels (depth)]
- ✓ RELU layer will apply an elementwise activation function, such as the $max(0,x)$
- ✓ Apply 24 6x6 filters to 12 layers (channel) with stride 2 - W[6,6,12,24] stride 2
- ✓ This may result in volume of [14x14x24] when we use 24 filters, stride 2.
- ✓ RELU layer will apply an elementwise activation function, such as the $max(0,x)$
- ✓ Apply 32 6x6 filters to 24 layers (channel) with stride 2 - W[6,6,24,32] stride 2
- ✓ This may result in volume of [7x7x32] when we use 32 filters, stride 2.
- ✓ RELU layer will apply an elementwise activation function, such as the $max(0,x)$
- ✓ Flatten the volume (tensor) to be a vector of 7x6x32=1568 elements
- ✓ Use 200 neurons with the input vector, Fully connected layer, W [1558, 200], output=1 x 200
- ✓ FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score.
- ✓ In this way, ConvNets transform the original image layer by layer from the original pixel values to the final class scores.

MLP – Fully NN Connected Model : MNIST

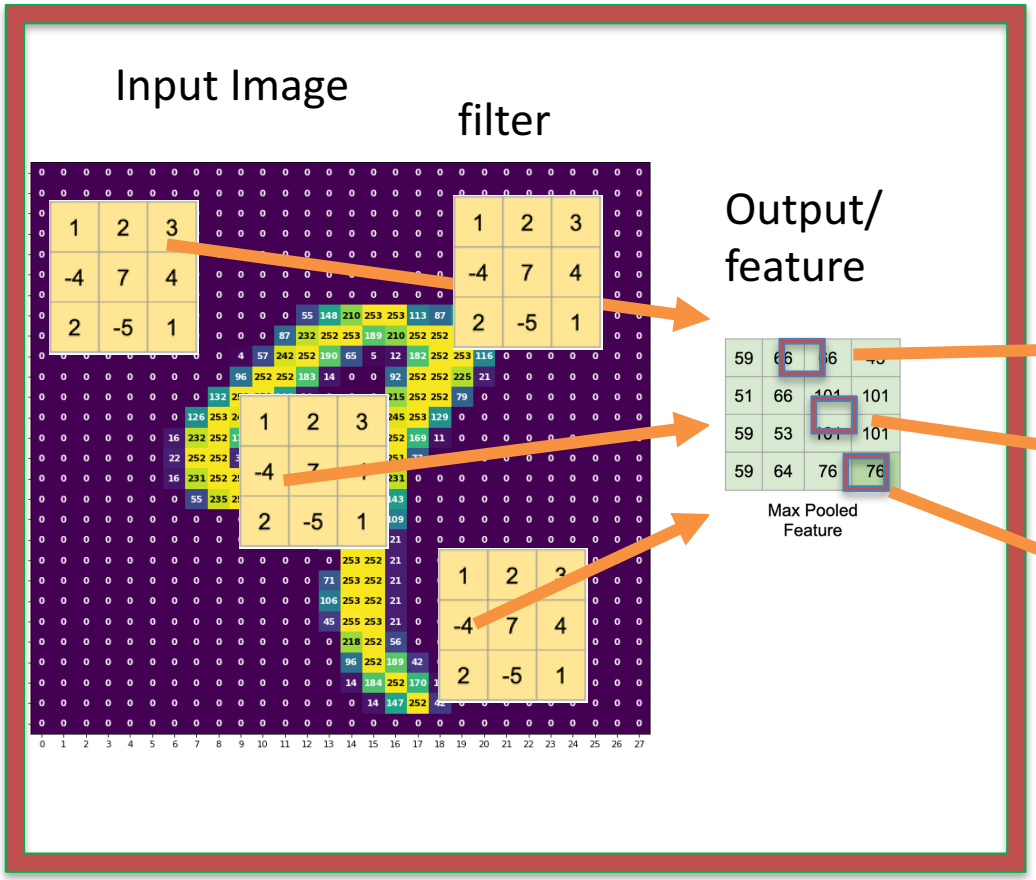
```
12
13 model = tf.keras.models.Sequential([
14   tf.keras.layers.Flatten(input_shape=(28, 28)),
15   tf.keras.layers.Dense(16, activation='relu'),
16   tf.keras.layers.Dense(16, activation='relu'),
17   tf.keras.layers.Dense(10,activation='softmax')
18 ])
```

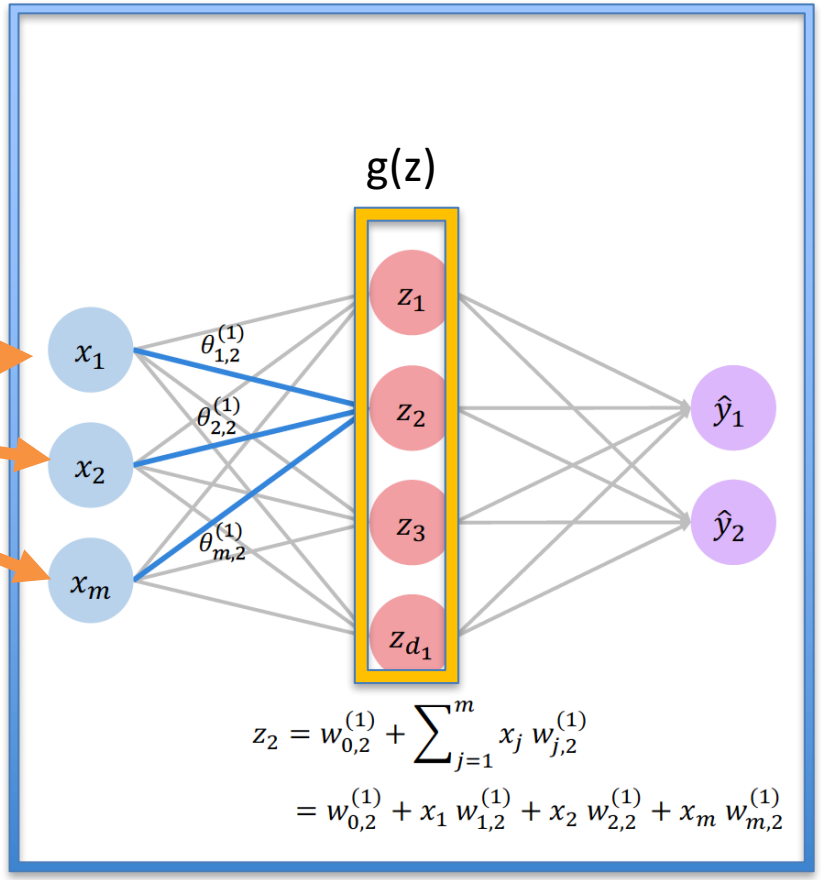
CNN – Convolutional NN Model : MNIST

```
54 model = keras.Sequential(
55     [
56         keras.Input(shape=input_shape),
57         layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
58         layers.MaxPooling2D(pool_size=(2, 2)),
59         layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
60         layers.MaxPooling2D(pool_size=(2, 2)),
61         layers.Flatten(),
62         layers.Dropout(0.5),
63         layers.Dense(num_classes, activation="softmax"),
64     ]
```

```
76 model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
77
78 model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_split=0.1)
```

Step 1: Parametric Model : Convolutional Neural Network (CNN)
Convolutional filter + Connected Neural Network

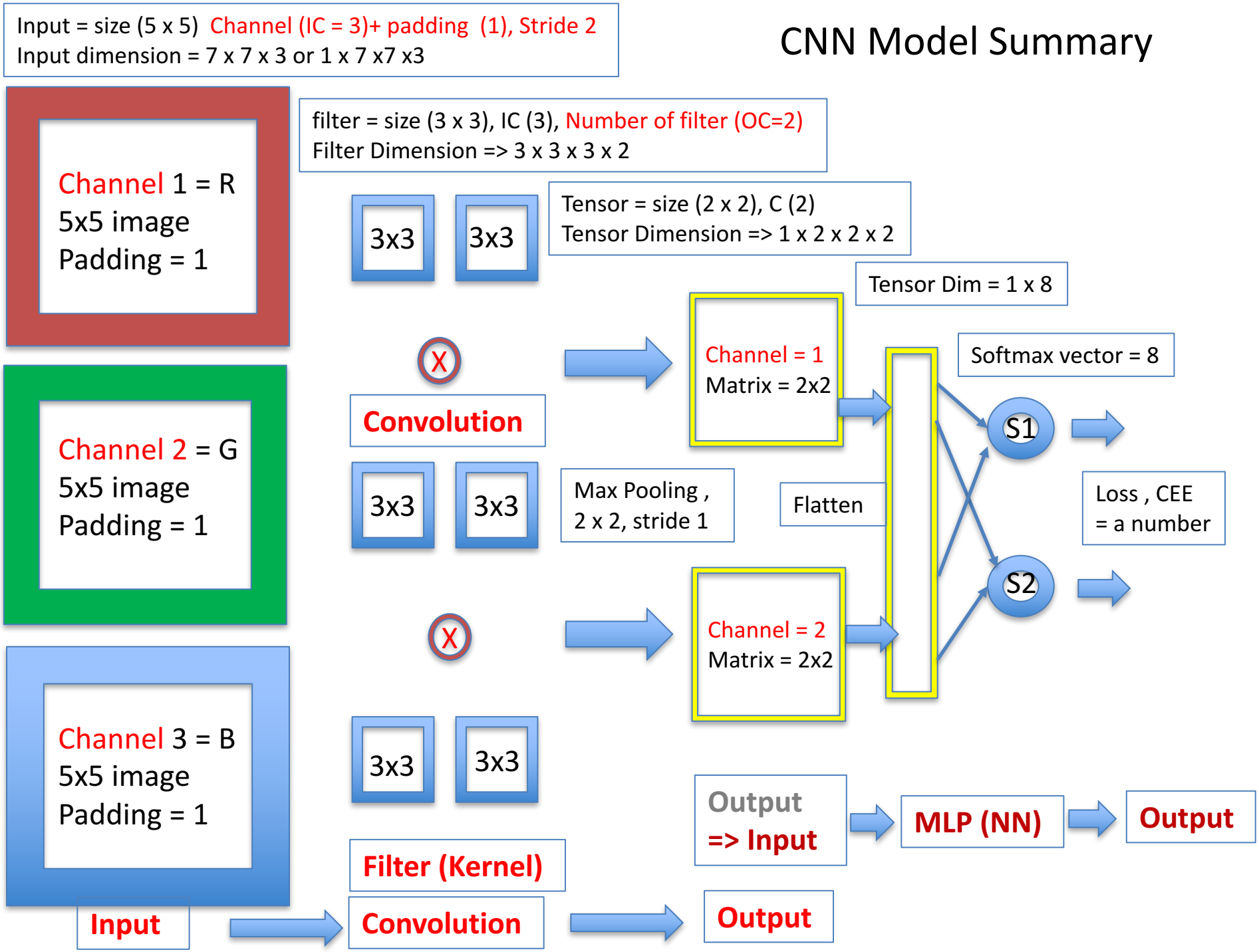




Convolutional filtering

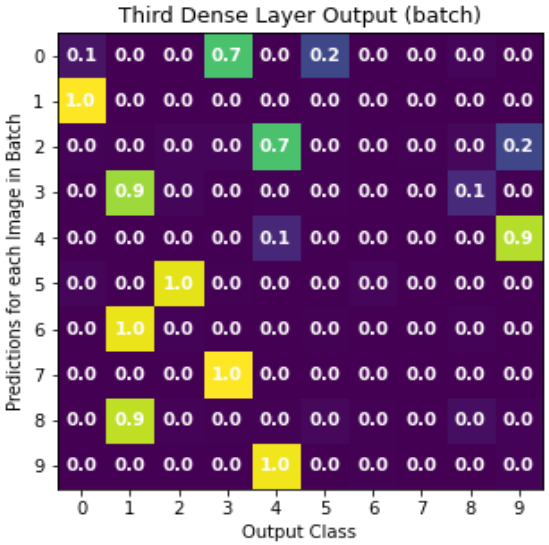
Multilayer Perceptron

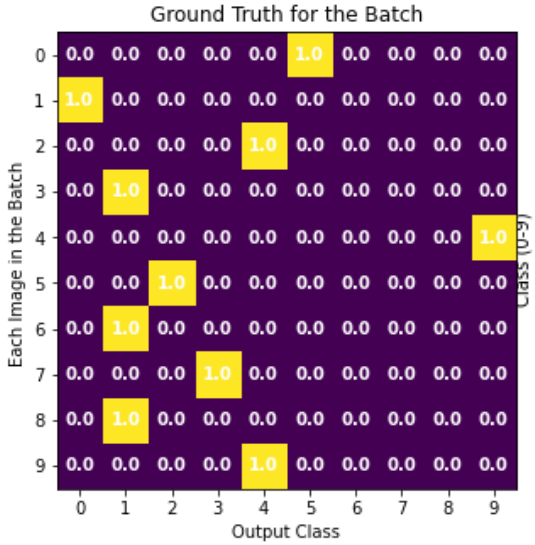
CNN Model Summary

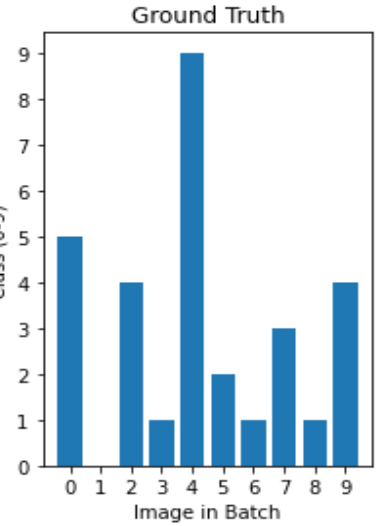


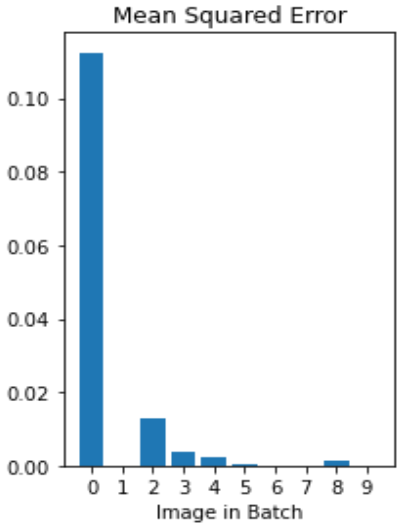
Evaluating the Error

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









Categorical Cross Entropy Loss

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{sj}}\right)$$

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ Softmax Function

$$L_i = -\log P(Y = y_i|X = x_i)$$

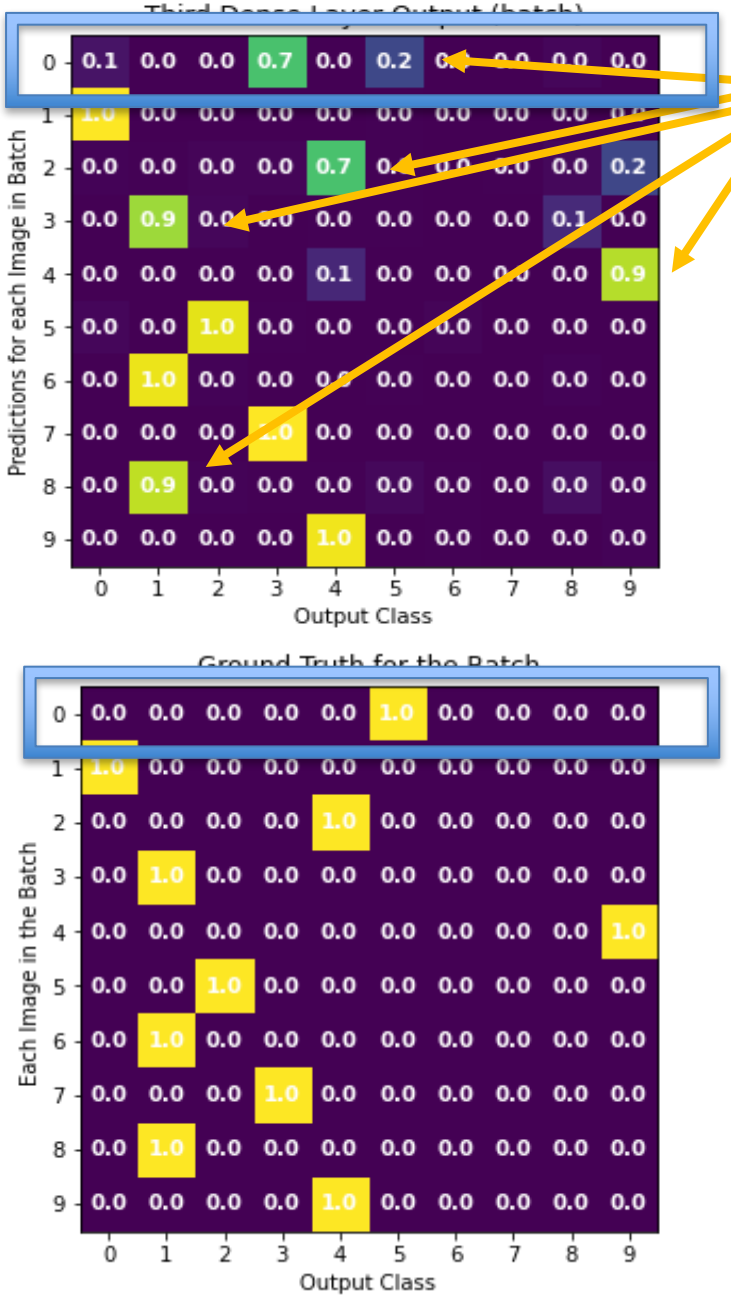
	Probabilities must be >= 0	exp	Probabilities must sum to 1	normalize		compare	
cat	**3.2**		**24.5**		**0.13**		**1.00**
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00
	Unnormalized log-probabilities / logits		unnormalized probabilities		probabilities		Correct probs

Cross Entropy

$$H(P, Q) = H(p) + D_{KL}(P\|Q)$$

L (i) = - log0.2 = 1.61
- log1.0 = 0
- log0.7 = 0.356
- log0.9 = 0.105
- log0.9 = 0.105
- log1.0 = 0
- log1.0 = 0
- log1.0 = 0
- log0.9 = 0.105
- log1.0 = 0

Evaluating the Error



Z

Categorical Cross Entropy Loss

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{sj}}\right)$$

L (i) = - log0.2 = 1.61	Image 0 = 5
- log1.0 = 0	Image 1 = 0
- log0.7 = 0.356	Image 2 = 4
- log0.9 = 0.105	Image 3 = 1
- log0.9 = 0.105	Image 4 = 9
- log1.0 = 0	Image 5 = 2
- log1.0 = 0	Image 6 = 1
- log1.0 = 0	Image 7 = 3
- log0.9 = 0.105	Image 8 = 2
- log1.0 = 0	Image 9 = 4

Training

"Training a neural network"

What does it mean?

What does the network train?

minimize error of the computed values against the labeled values (loss function)

What are the training parameters

W and b

How does it work?

optimization based on gradient descent algorithm

go back and forth in the NN model to adjust for w and b => need derivatives w.r.t. w and b

Go through forward calculation

training with backpropagation

DNN Forward Backward Calculation : Weights and bias

Algorithm

1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$

```
weights = tf.random_normal(shape, stddev=sigma)
```

2. Loop until convergence:
3. Compute gradient, $\frac{\partial J(W)}{\partial W}$

```
grads = tf.gradients(ys=loss, xs=weights)
```

4. Update weights, $W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$

```
weights_new = weights.assign(weights - lr * grads)
```

5. Return weights

In one epoch

1. Pick a mini-batch (32)
2. Feed it to Neural Network
3. Forward path calculation
4. Calculate the mean (1875 iteration) gradient of the mini-batch
5. Use the mean gradient calculated in step 4 to update the weights
6. Repeat steps 1–5 for the mini-batches we created

$$\frac{\sum_{j=1}^{m} \nabla C_{X_j}}{m} \approx \frac{\sum_x \nabla C_x}{n} = \nabla C,$$

$$w_k \to w_k' = w_k - \eta \frac{\partial C}{\partial w_k}$$

$$b_l \to b_l' = b_l - \eta \frac{\partial C}{\partial b_l}.$$

Tensorflow Example

The model is trained in about 10 seconds completing 5 epochs with a Nvidia GTX 1080 GPU and has an accuracy of around 97-98%

```
2020-07-29 19:53:40.592860: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1108]      0
2020-07-29 19:53:40.592871: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1121] 0:   N
2020-07-29 19:53:40.593073: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:981] successful NUMA node read from
SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2020-07-29 19:53:40.593535: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:981] successful NUMA node read from
SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2020-07-29 19:53:40.593973: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:981] successful NUMA node read from
SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2020-07-29 19:53:40.594387: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1247] Created TensorFlow device (/job:lo
calhost/replica:0/task:0/device:GPU:0 with 7219 MB memory) -> physical GPU (device: 0, name: GeForce GTX 1080, pci bus
id: 0000:26:00.0, compute capability: 6.1)
2020-07-29 19:53:40.623075: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x55d981198b80 initialized fo
r platform CUDA (this does not guarantee that XLA will be used). Devices:
2020-07-29 19:53:40.623096: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): GeForce GTX
1080, Compute Capability 6.1
2020-07-29 19:53:52.215159: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic
 library libcublas.so.10
Epoch 1/5
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2982 - accuracy: 0.9127
Epoch 2/5
1875/1875 [==============================] - 2s 1ms/step - loss: 0.1433 - accuracy: 0.9571
Epoch 3/5
1875/1875 [==============================] - 2s 1ms/step - loss: 0.1097 - accuracy: 0.9663
Epoch 4/5
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0893 - accuracy: 0.9730
Epoch 5/5
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0777 - accuracy: 0.9750
313/313 - 0s - loss: 0.0727 - accuracy: 0.9776
```

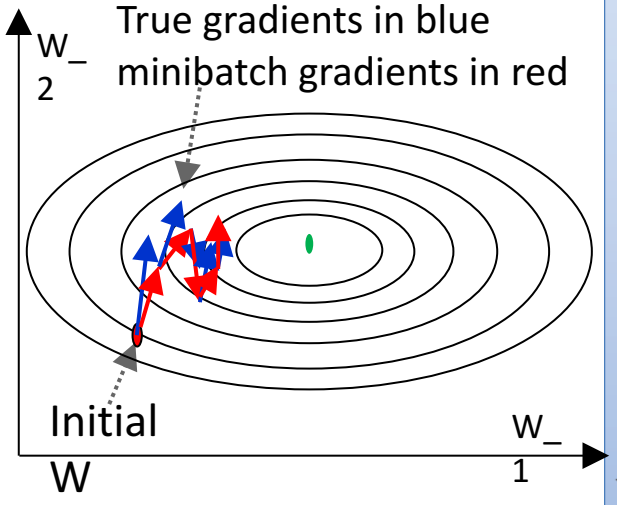
Batch = 60000/1875 = 32, exhaust it => 1 epoch

Optimization: Batch SGD

Instead of computing a gradient across the entire dataset with size N, divides the dataset into a fixed-size subset ("minibatch") with size m, and computes the gradient for each update on this smaller batch:

(N is the dataset size, m is the minibatch size)

$$\mathbf{g} = \frac{1}{m'} \sum_{i=1}^{m'} \nabla_{\boldsymbol{\theta}} E_i(\mathbf{x}^{(n_i)}, y^{(n_i)}, \boldsymbol{\theta}),$$
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \mathbf{g},$$

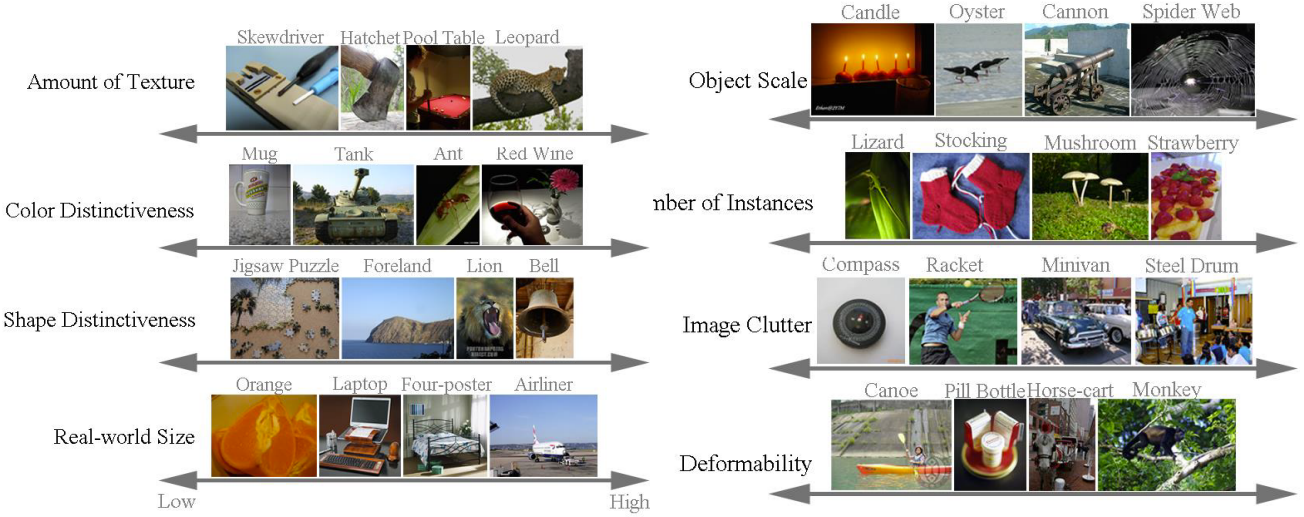


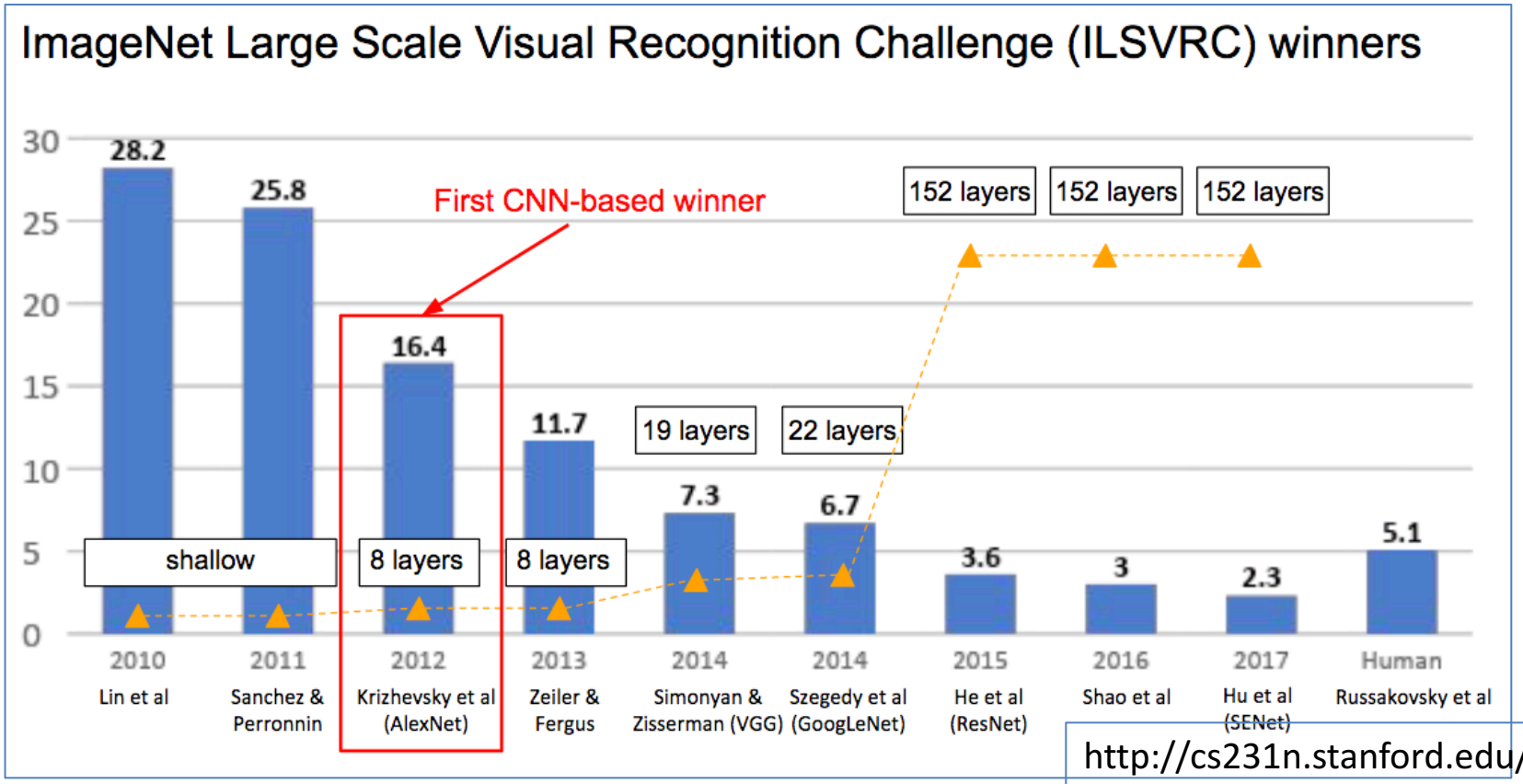
Given: training set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
Initialize all $\Theta^{(l)}$ randomly (NOT to 0!)
Loop // each iteration is called an epoch
  Loop // each iteration is a mini-batch
    Set $\Delta_{i,j}^{(l)} = 0 \quad \forall l, i, j$ (Used to accumulate gradient)
    Sample m training instances $\mathcal{X} = \{(\mathbf{x}'_1, y'_1), \ldots, (\mathbf{x}'_m, y'_m)\}$ without replacement
    For each instance in $\mathcal{X}$, $(\mathbf{x}_k, y_k)$:
      Set $\mathbf{a}^{(1)} = \mathbf{x}_k$
      Compute $\{\mathbf{a}^{(2)}, \ldots, \mathbf{a}^{(L)}\}$ via forward propagation
      Compute $\boldsymbol{\delta}^{(L)} = \mathbf{a}^{(L)} - y_k$
      Compute errors $\{\boldsymbol{\delta}^{(L-1)}, \ldots, \boldsymbol{\delta}^{(2)}\}$
      Compute gradients $\Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$
    Compute mini-batch regularized gradient $D_{ij}^{(l)} = \begin{cases} \frac{1}{m}\Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} & \text{if } j \neq 0 \\ \frac{1}{m}\Delta_{ij}^{(l)} & \text{otherwise} \end{cases}$
    Update weights via gradient step $\Theta_{ij}^{(l)} = \Theta_{ij}^{(l)} - \alpha D_{ij}^{(l)}$
  Until all training instances are seen
Until weights converge or max #epochs is reached

1875 TIMES OF BATCH ITERATION

BATCH LOOP

- ✓ 1000 classes
- ✓ RGB images
- ✓ Cluttered Images (one category)
- ✓ Full resolution images
- ✓ 1.2 million training images
- ✓ 100 thousand test images





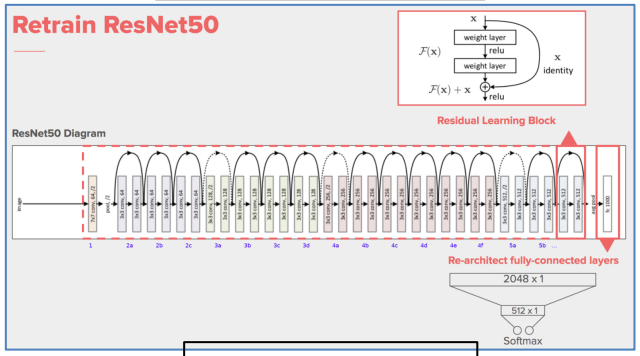
http://cs231n.stanford.edu/

Neural Network Models

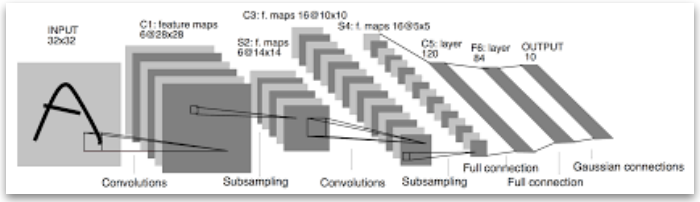
Dataset → Pre-defined Neural Network Model → Fully connected layer # neurons = # classes → Classification

Examples of Pre-defined Neural Network Models

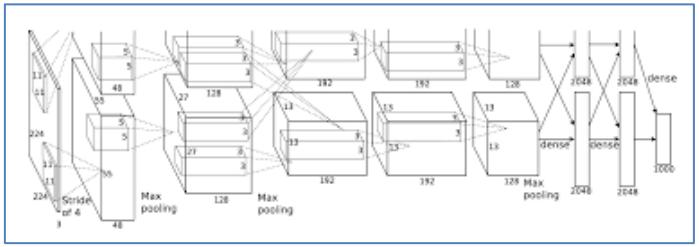
ResNet50



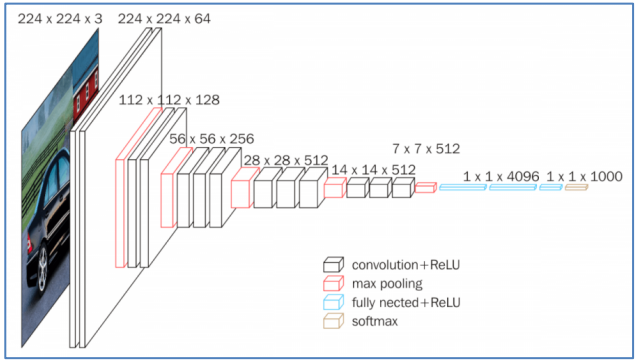
LeNet5



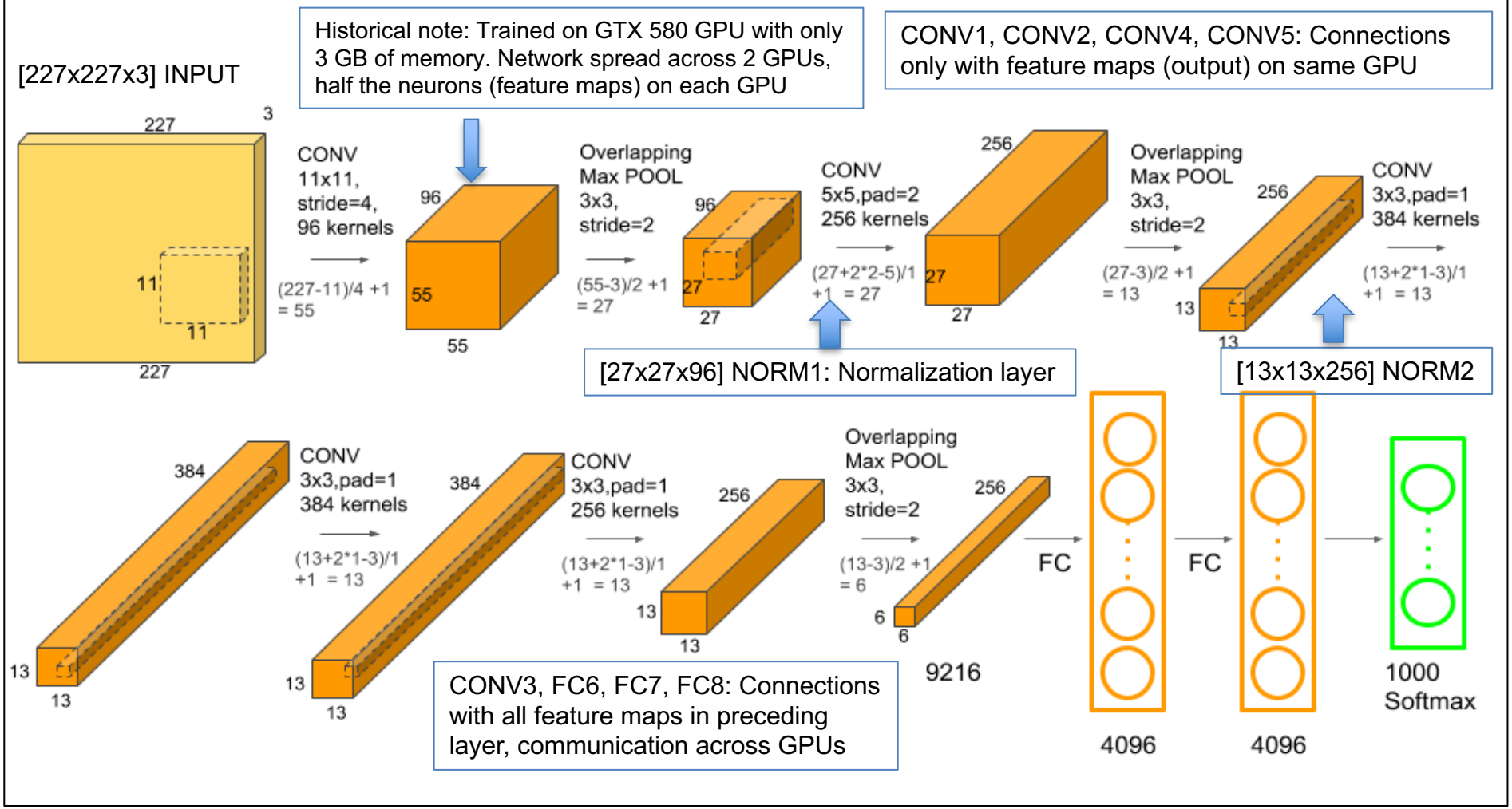
AlexNet



VGG16

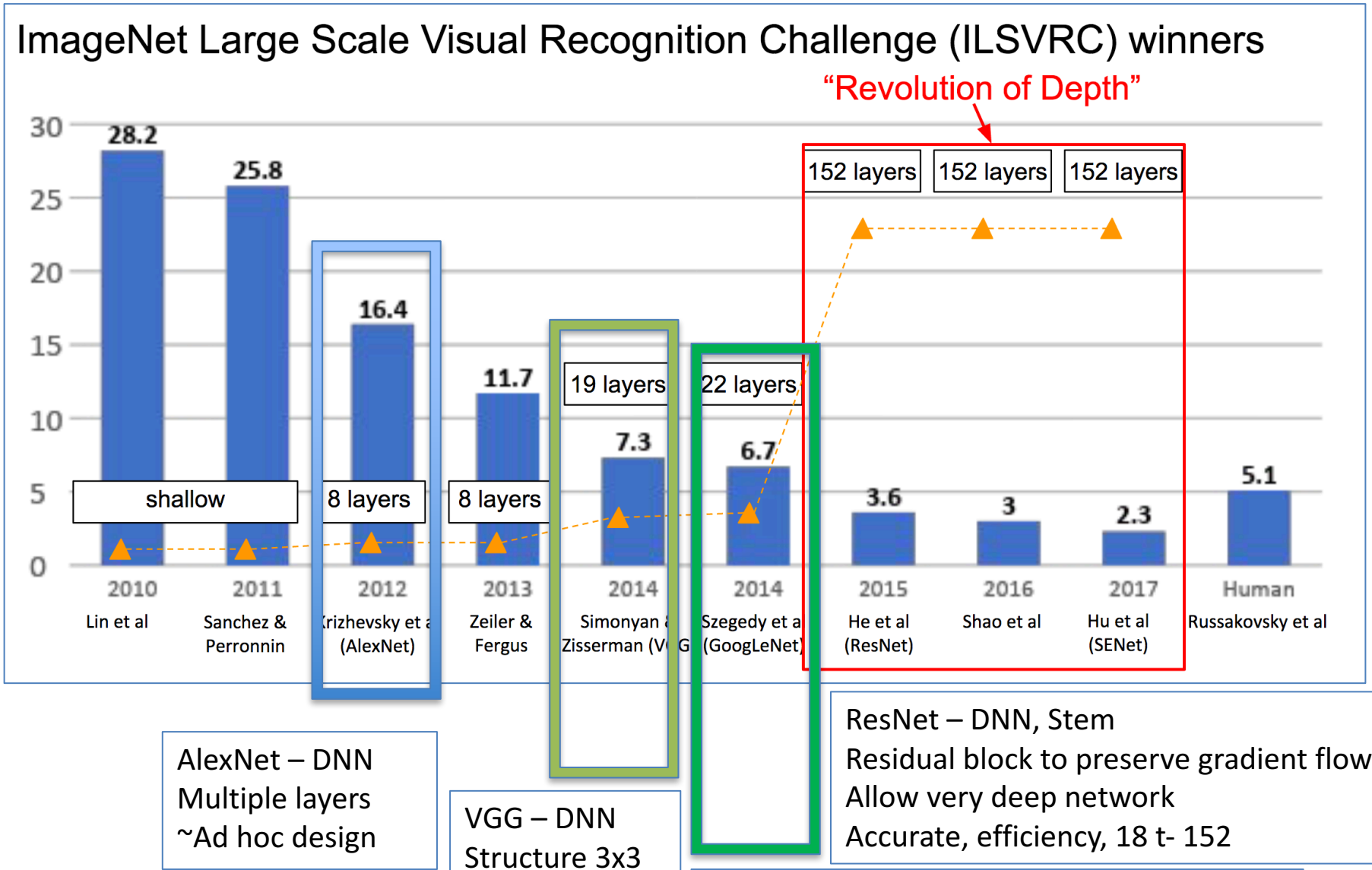


AlexNet architecture, Training



Details/Retrospectives: - (1) first use of ReLU, (2) used Norm layers (not common anymore), (3) heavy data augmentation, (4) dropout 0.5, (5) batch size 128, (6) SGD Momentum 0.9, (7) Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus, (8) L2 weight decay 5e-4, (9) 7 CNN ensemble: 18.2% -> 15.4%

https://www.learnopencv.com/number-of-parameters-and-tensor-sizes-in-convolutional-neural-network/



AlexNet – DNN
Multiple layers
~Ad hoc design

VGG – DNN
Structure 3x3
Deep NN,
Lots of parameters
Accuracy

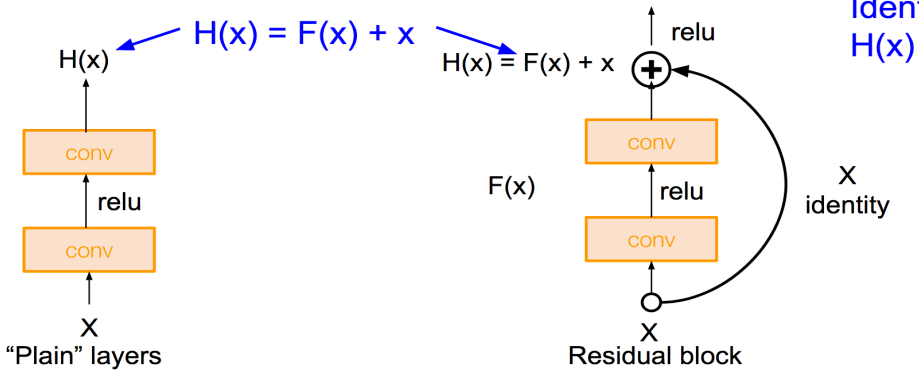
GoogleNet – DNN
Stem to reduce feature size
Block structure for efficiency
Group pooling to reduce parameter count

ResNet – DNN, Stem
Residual block to preserve gradient flow
Allow very deep network
Accurate, efficiency, 18 t- 152

http://cs231n.stanford.edu/

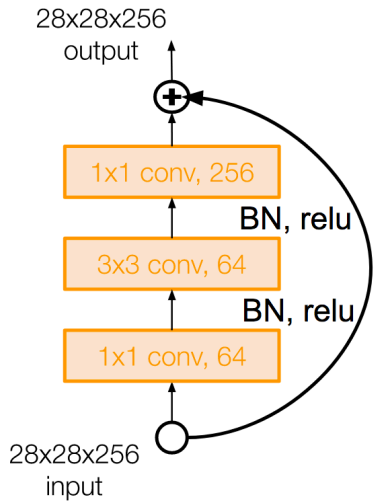
ResNet

Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping



Identity mapping:
H(x) = x if F(x) = 0

Use layers to fit **residual** F(x) = H(x) - x instead of H(x) directly



For deeper networks (ResNet-50+), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)

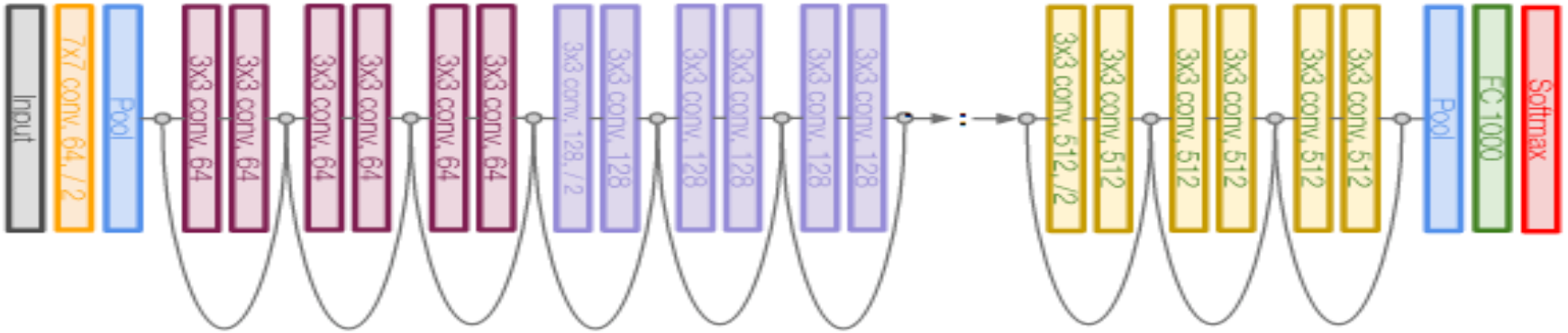
- ✓ Full ResNet architecture:-
- ✓ Stack residual blocks
- ✓ Every residual block has two 3x3 conv layers
- ✓ Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- ✓ Additional conv layer at the beginning (stem)
- ✓ No FC layers at the end (only FC 1000 to output classes)

Total depths of 18, 34, 50, 101, or 152 layers for ImageNet

1x1 conv, 256 filters projects back to 256 feature maps (28x28x256)

3x3 conv operates over only 64 feature maps

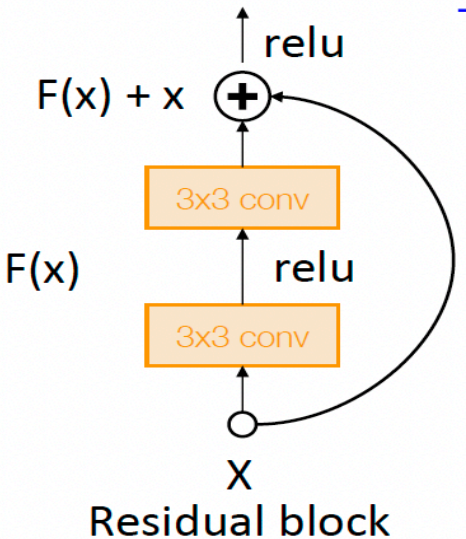
1x1 conv, 64 filters to project to 28x28x64

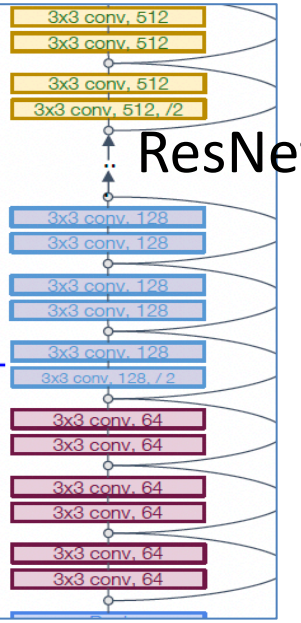


A residual network is a stack of many residual blocks

Regular design, like VGG: each residual block has two 3x3 conv

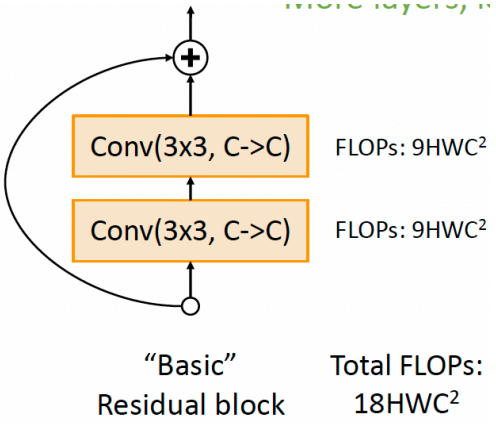
Network is divided into **stages**: the first block of each stage halves the resolution (with stride-2 conv) and doubles the number of channels





ResNet

Basic Block
ResNet18, 34



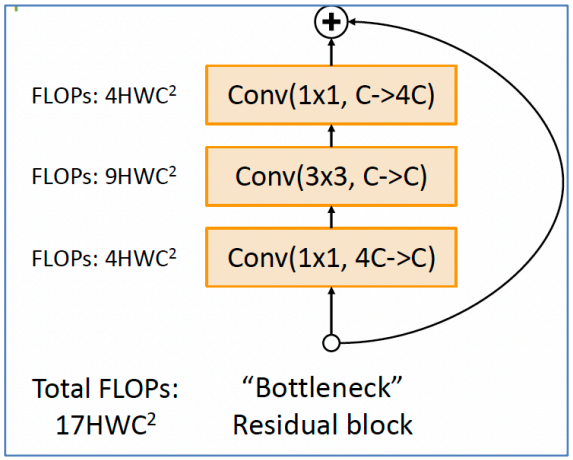
ResNet-18:
Stem: 1 conv layer
Stage 1 (C=64): 2 res. block = 4 conv
Stage 2 (C=128): 2 res. block = 4 conv
Stage 3 (C=256): 2 res. block = 4 conv
Stage 4 (C=512): 2 res. block = 4 conv
Linear

ImageNet top-5 error: 10.92
GFLOP: 1.8

ResNet-34:
Stem: 1 conv layer
Stage 1: 3 res. block = 6 conv
Stage 2: 4 res. block = 8 conv
Stage 3: 6 res. block = 12 conv
Stage 4: 3 res. block = 6 conv
Linear

ImageNet top-5 error: 8.58
GFLOP: 3.6

Bottleneck Block
ResNet50 …



From Now to Beyond

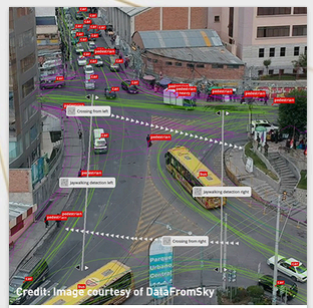
From predict to build..
From modeling to generating
From control to autonomous
From composition to integration
From predict to build…

From Now to Beyond

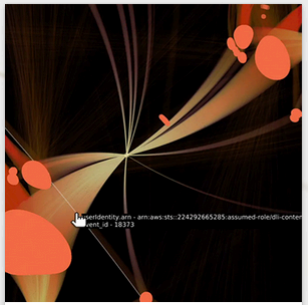
Keynote

Accelerated Computing and Dev Tools

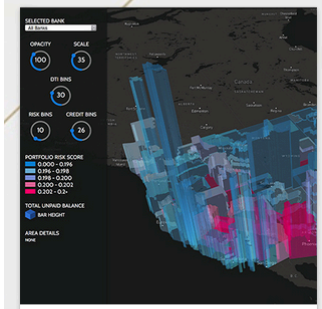
Autonomous Vehicles



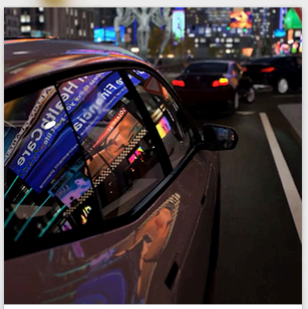
Computer Vision/ Video Analytics



Cybersecurity

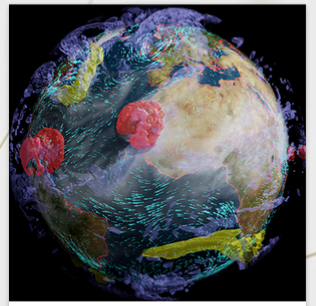


Data Science

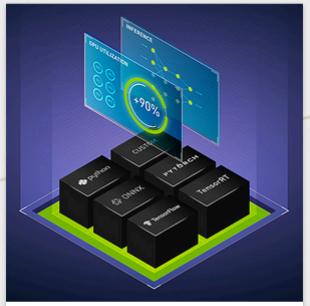


Game Development

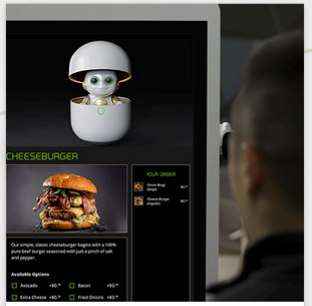
Graphics, Design Collaboration, and Digital Twins



HPC



Inference

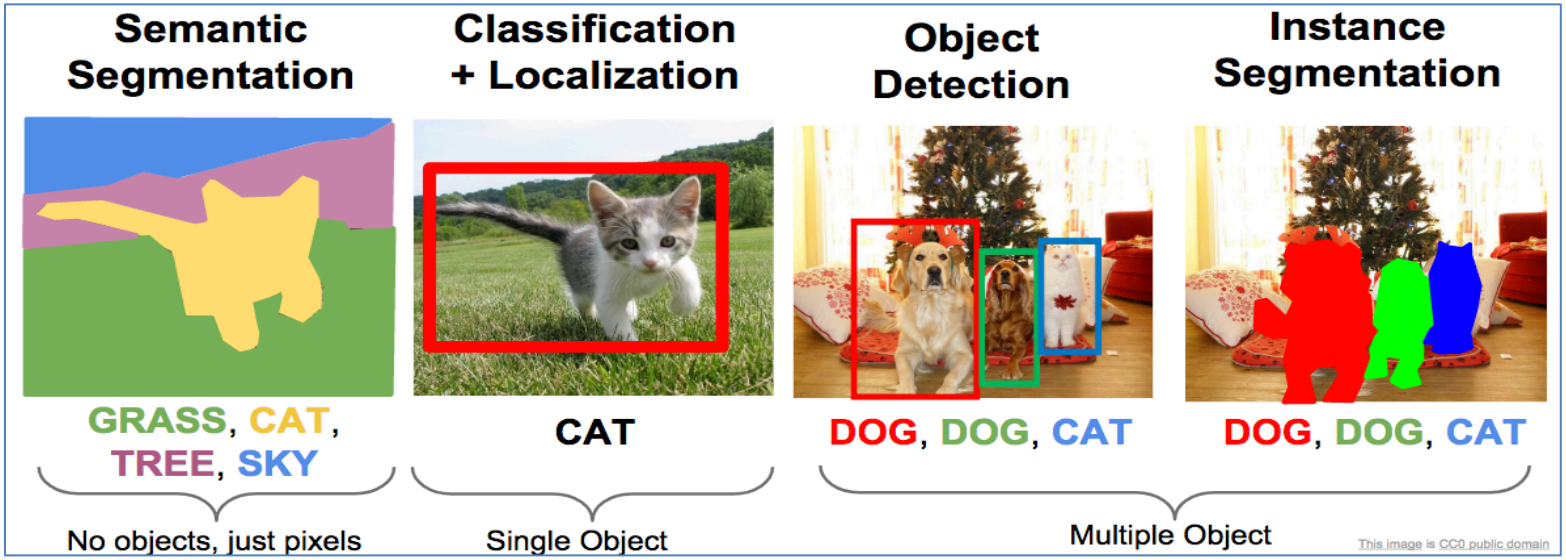


Recommender System

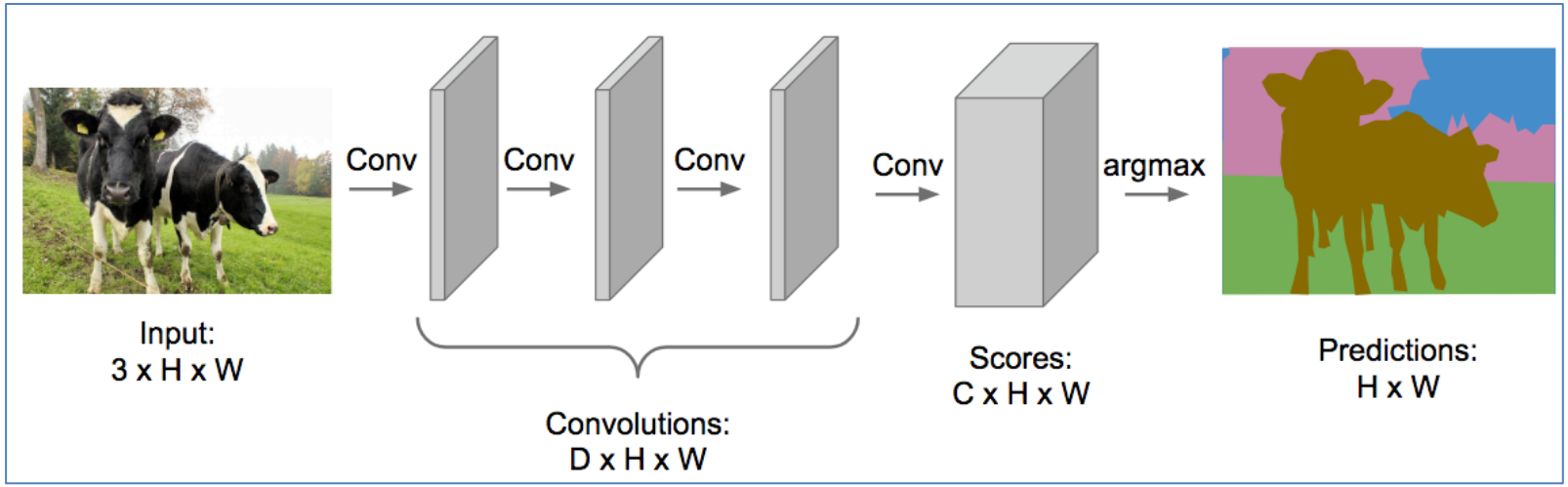
Robotics

Speech AI / NLP

Segmentation and Detection



Segmentation : Label each pixel in the image with a category label



https://github.com/tensorflow/hub/blob/master/examples/colab/tf2_object_detection.ipynb

http://cs231n.stanford.edu/slides/2020/lecture_12.pdf

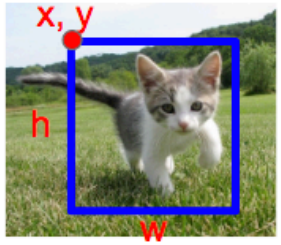
Single Object Detection

Used weighted sum of Losses of Softmax (CE) and L2(MSE) (scalar)

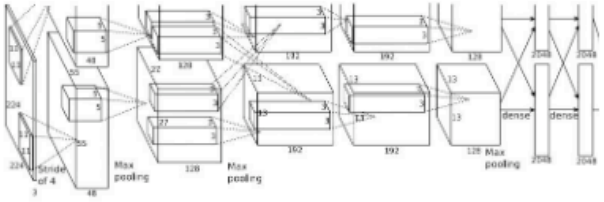
Object Detection: Single Object

(Classification + Localization)

Use pretrain network
Transfer learning from
Imagenet dataset



This image is CC0 public domain



Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label: Cat

Softmax Loss

Multitask Loss

Loss

Vector: 4096

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

Correct box: (x', y', w', h')

Treat localization as a regression problem!

http://cs231n.stanford.edu/slides/2020/lecture_12.pdf

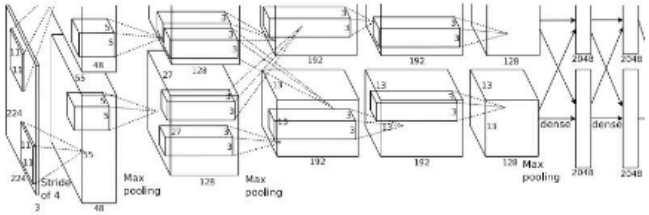
https://github.com/tensorflow/hub/blob/master/examples/colab/tf2_object_detection.ipynb

https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html

Object Detection: Multiple Objects Using sliding Windows

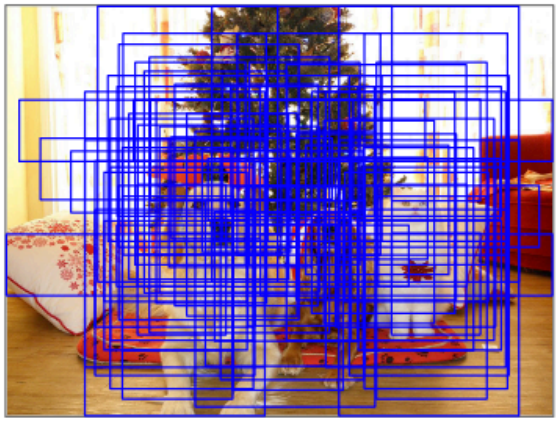
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Slide window region -> many inputs -> classification

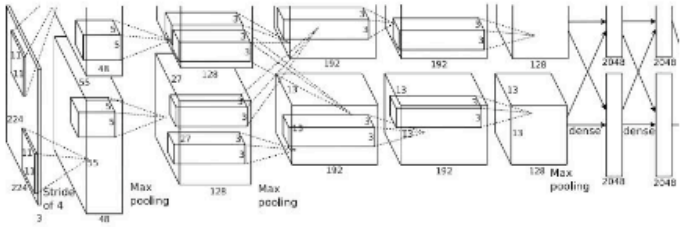


Dog? YES
Cat? NO
Background? NO

https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html



Slide window region -> too expensive, NO

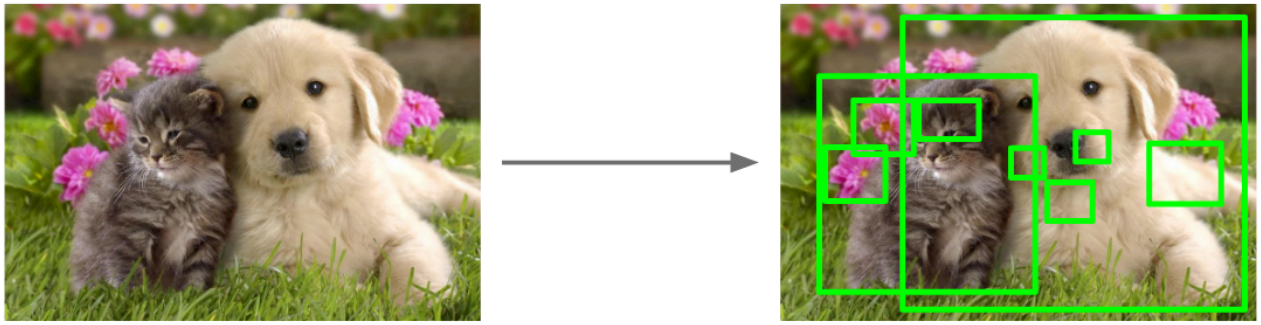


Dog? NO
Cat? YES
Background? NO

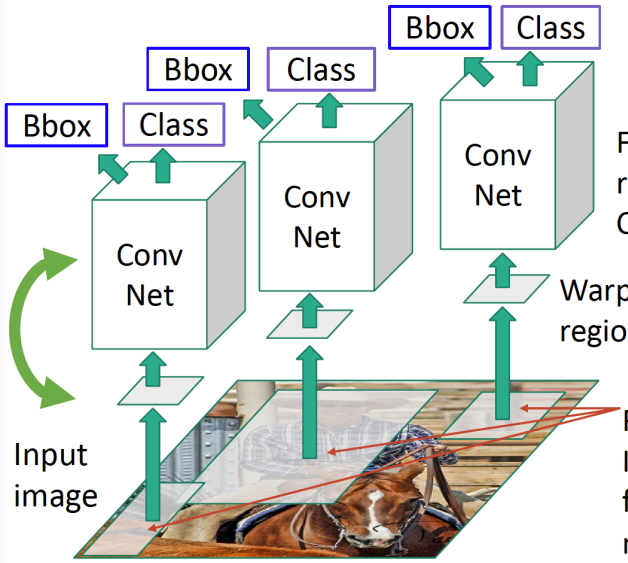
Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

Multiple Object Detection
Region Proposals : Selective Search

- Find "blobby" image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 1000 region proposals in a few seconds on CPU



R-CNN: Region-Based CNN



Classify each region

Bounding box regression: Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

Forward each region through ConvNet

Warped image regions (224x224)

Regions of Interest (RoI) from a proposal method (~2k)

Problem: Very slow! Need to do ~2k forward passes for each image!

Solution: Run CNN *before* warping!

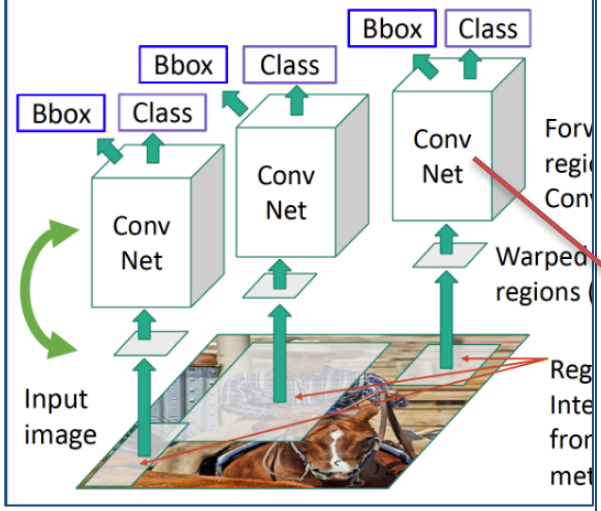
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

1. Run region proposal method to compute ~2000 region proposals
2. Resize each region to 224x224 and run independently through CNN to predict class scores and bbox transform
3. Use scores to select a subset of region proposals to output
4. (Many choices here: threshold on background, or per-category? Or take top K proposals per image?)
5. Compare with ground-truth boxes
6. Too Slow to compute 2000 CNN

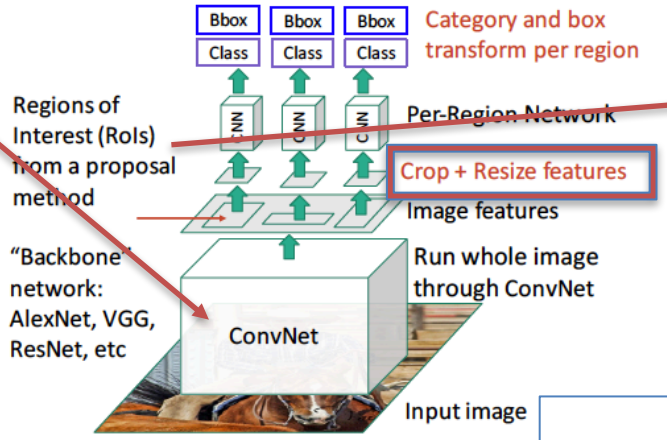
https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html

Object Detection

R-CNN: Region-Based CNN

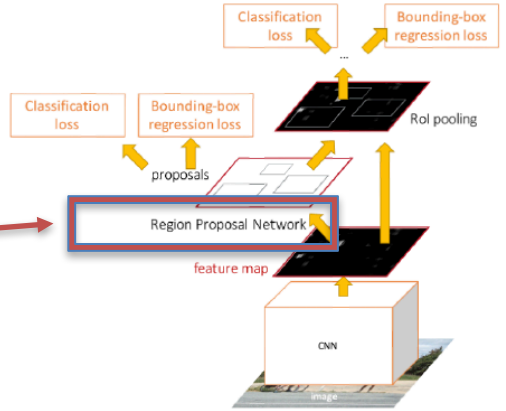


Fast R-CNN: Apply differentiable cropping to shared image features

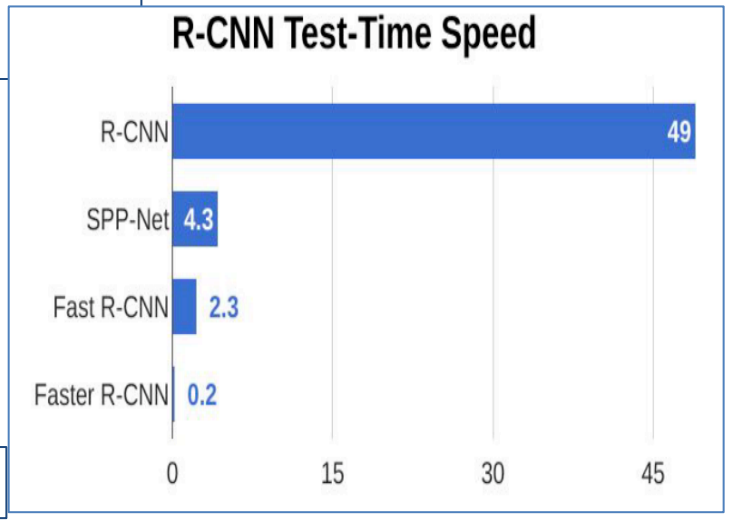


Faster R-CNN: Compute proposals with CNN

Replace Selective Search with CNN



TensorFlow Detection API:
https://github.com/tensorflow/models/tree/master/research/object_detection
Faster R-CNN, SSD, RFCN, Mask R-CNN
Detectron2 (PyTorch):
https://github.com/facebookresearch/detectron2
Fast / Faster / Mask R-CNN, RetinaNet, ImageAI..

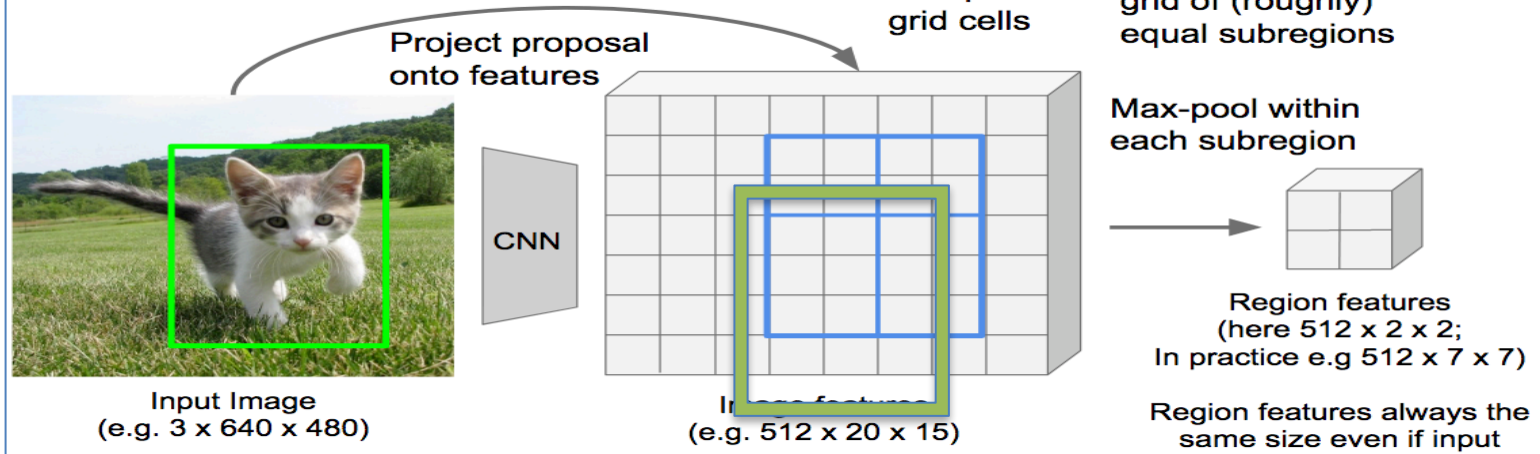


Input -> CONV -> Multi-task -stage -> Loss (CE+MSE+..) -> output

https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html

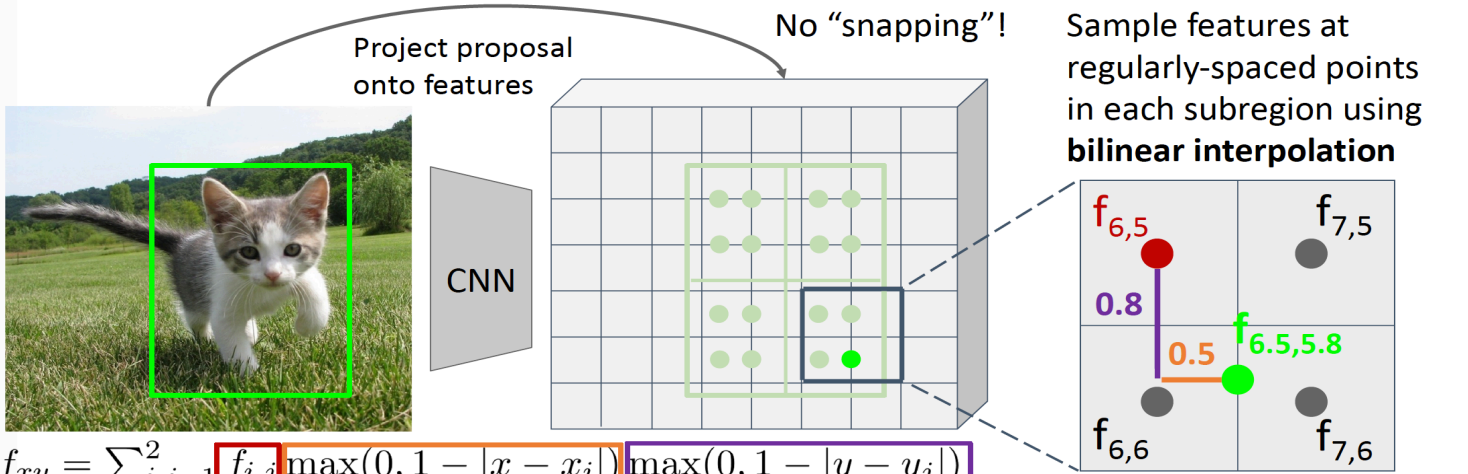
Cropping : Region of Interest

Cropping Features: RoI Pool



Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Align



$$f_{xy} = \sum_{i,j=1}^{2} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_j|)$$

$f_{6.5,5.8}$ = ($f_{6,5}$ * 0.5 * 0.2) + ($f_{7,5}$ * 0.5 * 0.2)
+ ($f_{6,6}$ * 0.5 * 0.8) + ($f_{7,6}$ * 0.5 * 0.8)

Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

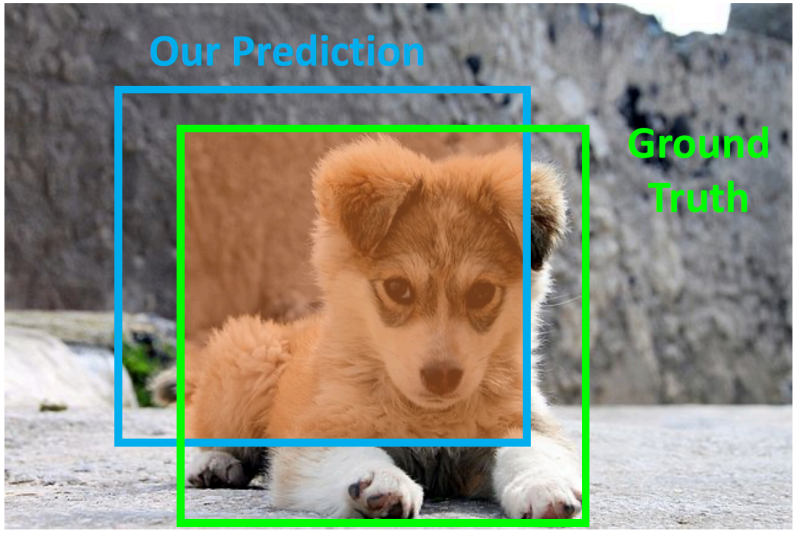
https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html

Performance Evaluation:
Comparing Boxes: Intersection over Union (IoU)

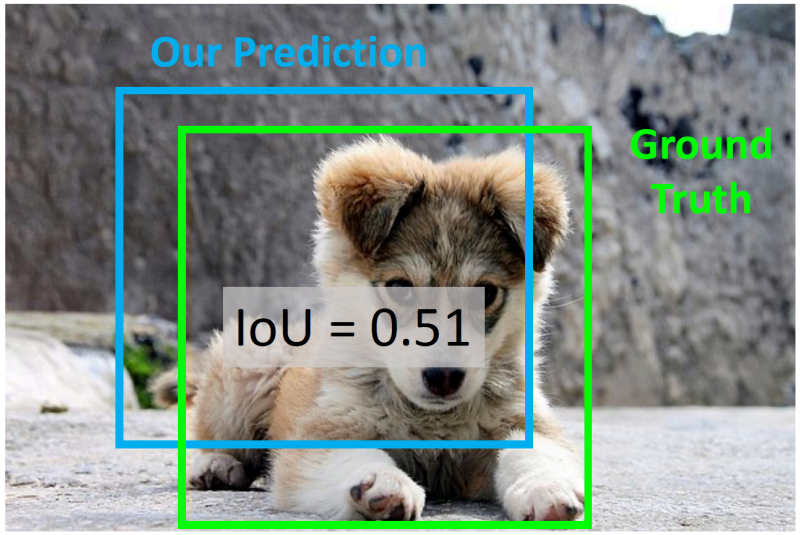
How can we compare our prediction to the ground-truth box?

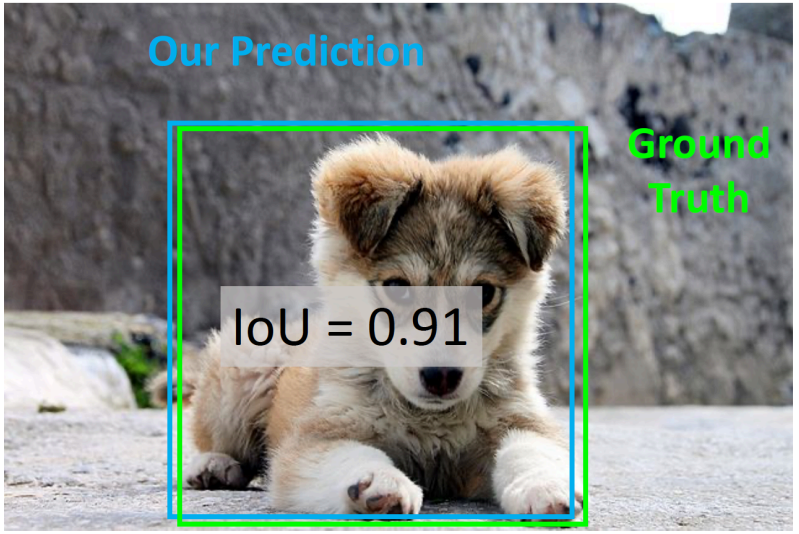
Intersection over Union (IoU)
(Also called “Jaccard similarity” or “Jaccard index”):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

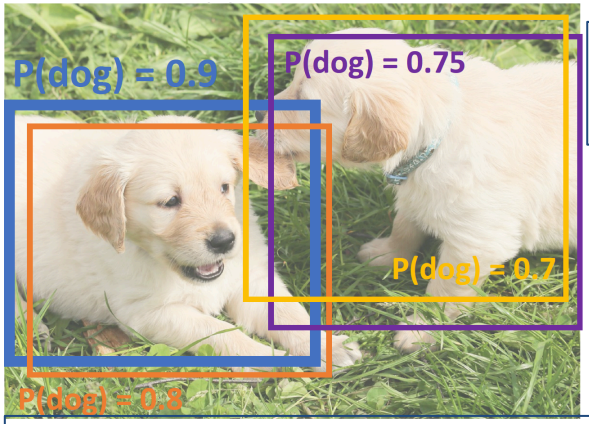


https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html





IoU > 0.5 is “decent”, IoU > 0.7 is “pretty good”, IoU > 0.9 is “almost perfect”

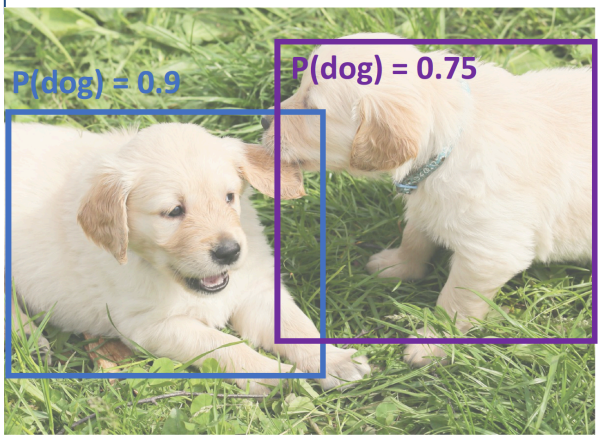


Overlapping Boxes: Non-Max Suppression (NMS)

Problem: NMS may eliminate "good" boxes when objects are highly overlapping… no good solution

Problem: Object detectors often output many overlapping detections:
Solution: Post-process raw detections using Non-Max Suppression (NMS)
1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

P(dog) = 0.8
P(dog) = 0.75
P(dog) = 0.7
IoU(■, ■) = 0.78
IoU(■, ■) = 0.05
IoU(■, ■) = 0.07



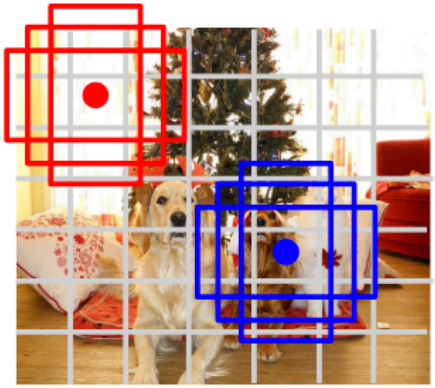
Evaluating Object Detectors: Mean Average Precision (mAP)

mAP@0.5 = 0.77
mAP@0.55 = 0.71
mAP@0.60 = 0.65
…
mAP@0.95 = 0.2
COCO mAP = 0.4

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve
   2. Average Precision (AP) = area under PR curve
   3. Mean Average Precision (mAP) = average of AP for each category
   4. For "COCO mAP": Compute mAP@thresh for each IoU threshold (0.5, 0.55, 0.6, …, 0.95) and take average

YOLO/SSD

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes** centered at each grid cell
Here B = 3

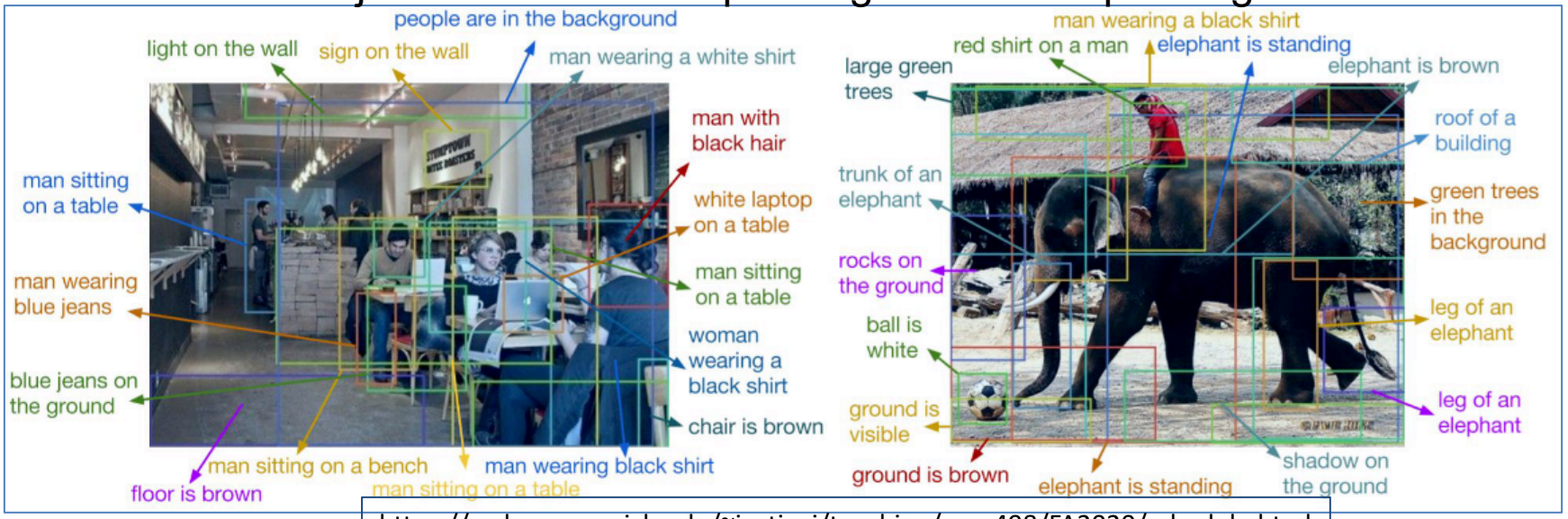
Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)
- Looks a lot like RPN, but category-specific!

Output:
7 x 7 x (5 * B + C)

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016
Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

Object Detection + Captioning= Dense Captioning



https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html

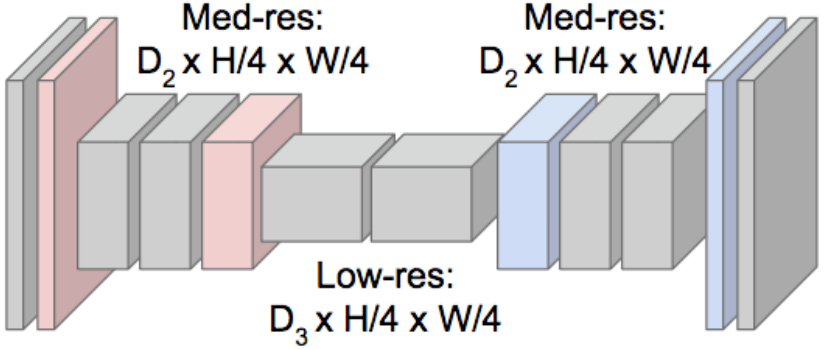
Semantic Segmentation Idea: Fully Convolutional

Downsampling: Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling: Unpooling or strided transpose convolution

Input:
3 x H x W



High-res:
D1 x H/2 x W/2

High-res:
D1 x H/2 x W/2

Predictions:
H x W

Max Pooling
Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4

5	6
7	8

Output: 2 x 2

Rest of the network

Max Unpooling
Use positions from pooling layer

1	2
3	4

Input: 2 x 2

0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

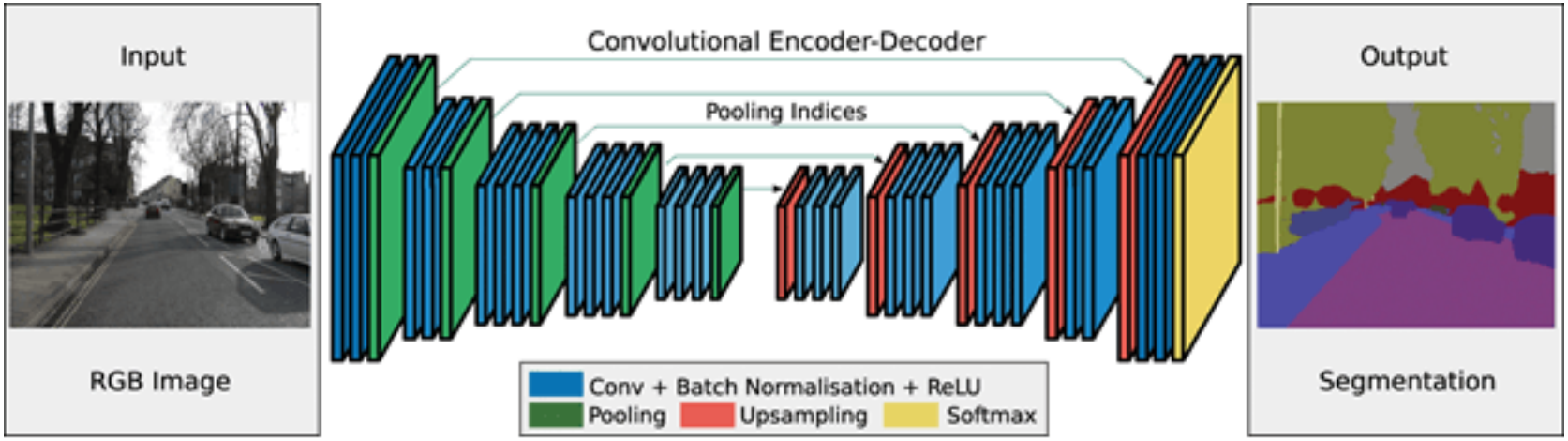
Output: 4 x 4

Image segmentation

Divide an image into multiple segments, every pixel in an image is associated with an object, instance segmentation and semantic segmentation :

In semantic segmentation, all objects of the same type are marked using one class label while in instance segmentation similar objects get their own separate labels.

The basic architecture in image segmentation consists of an encoder and a decoder.

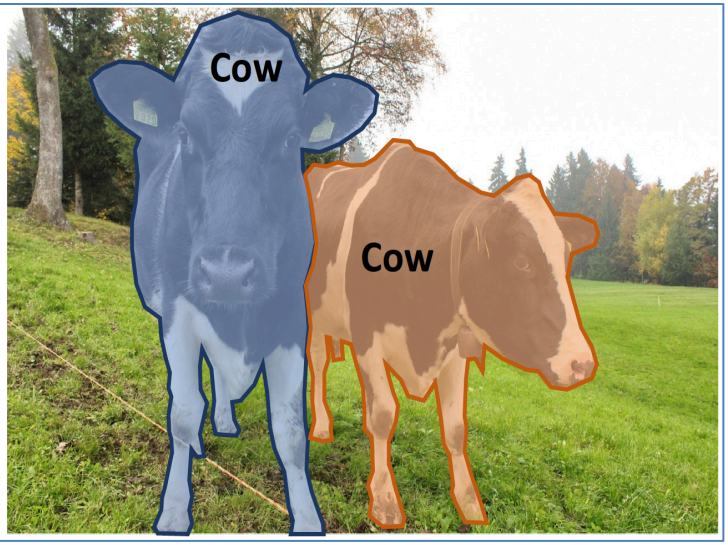


The encoder extracts features from the image through filters. The decoder is responsible for generating the final output which is usually a segmentation mask containing the outline of the object. Most of the architectures have this architecture or a variant of it.

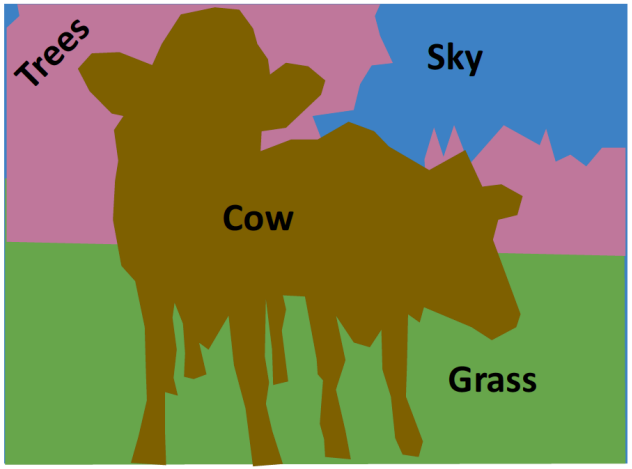
https://neptune.ai/blog/image-segmentation-in-2020

Instance Segmentation: Detect all objects in the image, and identify the pixels that belong to each object (Only things!)

Approach: Perform object detection, then predict a segmentation mask for each object!



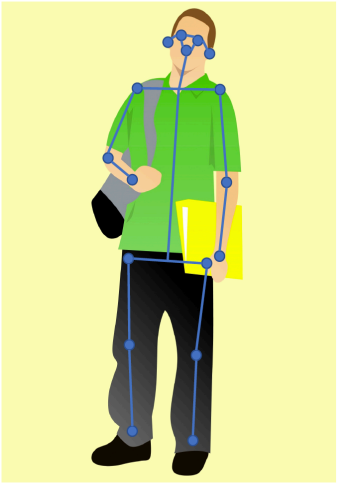
Semantic Segmentation: Identify both things and stuff, but doesn't separate instances



Beyond Instance Segmentation: Human Keypoints

Represent the pose of a human by locating a set of **keypoints**

e.g. 17 keypoints:
- Nose
- Left / Right eye
- Left / Right ear
- Left / Right shoulder
- Left / Right elbow
- Left / Right wrist
- Left / Right hip
- Left / Right knee
- Left / Right ankle

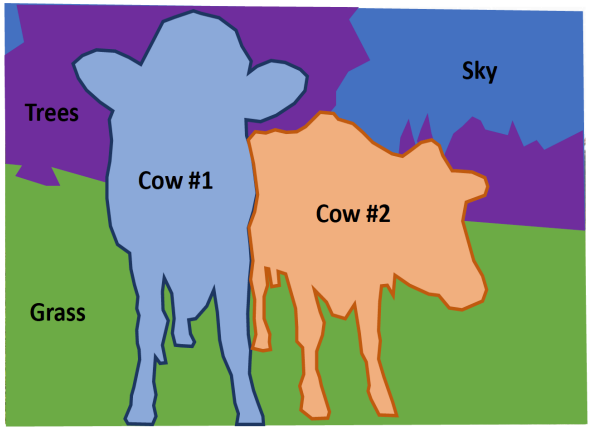


CC0 public domain

Beyond Instance Segmentation: **Panoptic Segmentation**

Label all pixels in the image (both things and stuff)

For "thing" categories also separate into instances



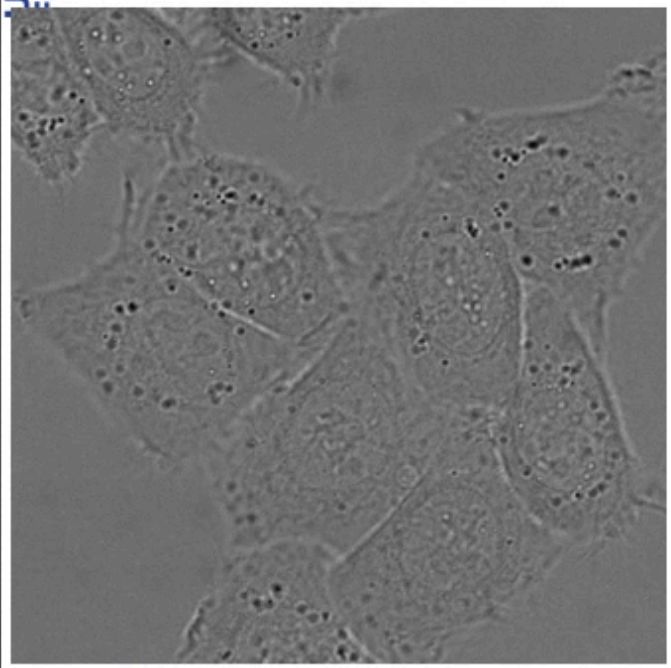
Kirillov et al, "Panoptic Segmentation", CVPR 2019
Kirillov et al, "Panoptic Feature Pyramid Networks", CVPR 2019

U-net: Convolutional Networks for Biomedical Image Segmentation

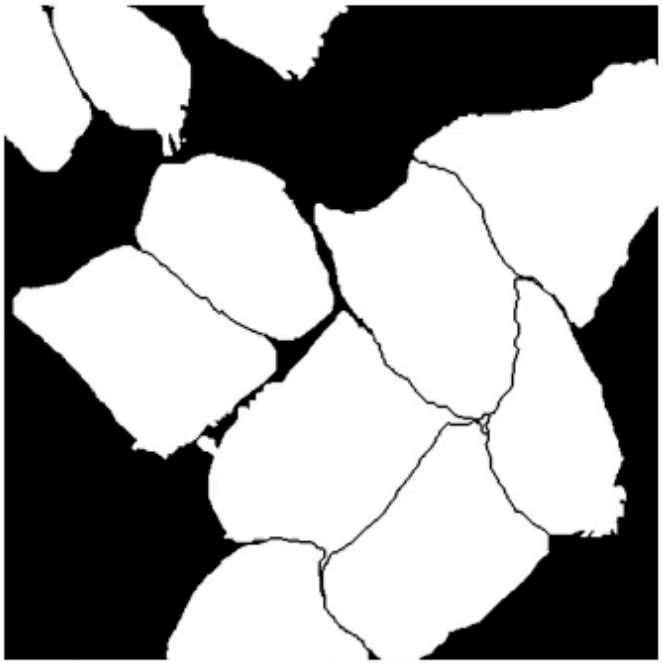
Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and
BIOSS Centre for Biological Signalling Studies
University of Freiburg, Germany

Biomedical Image Segmentation with U-net



input image

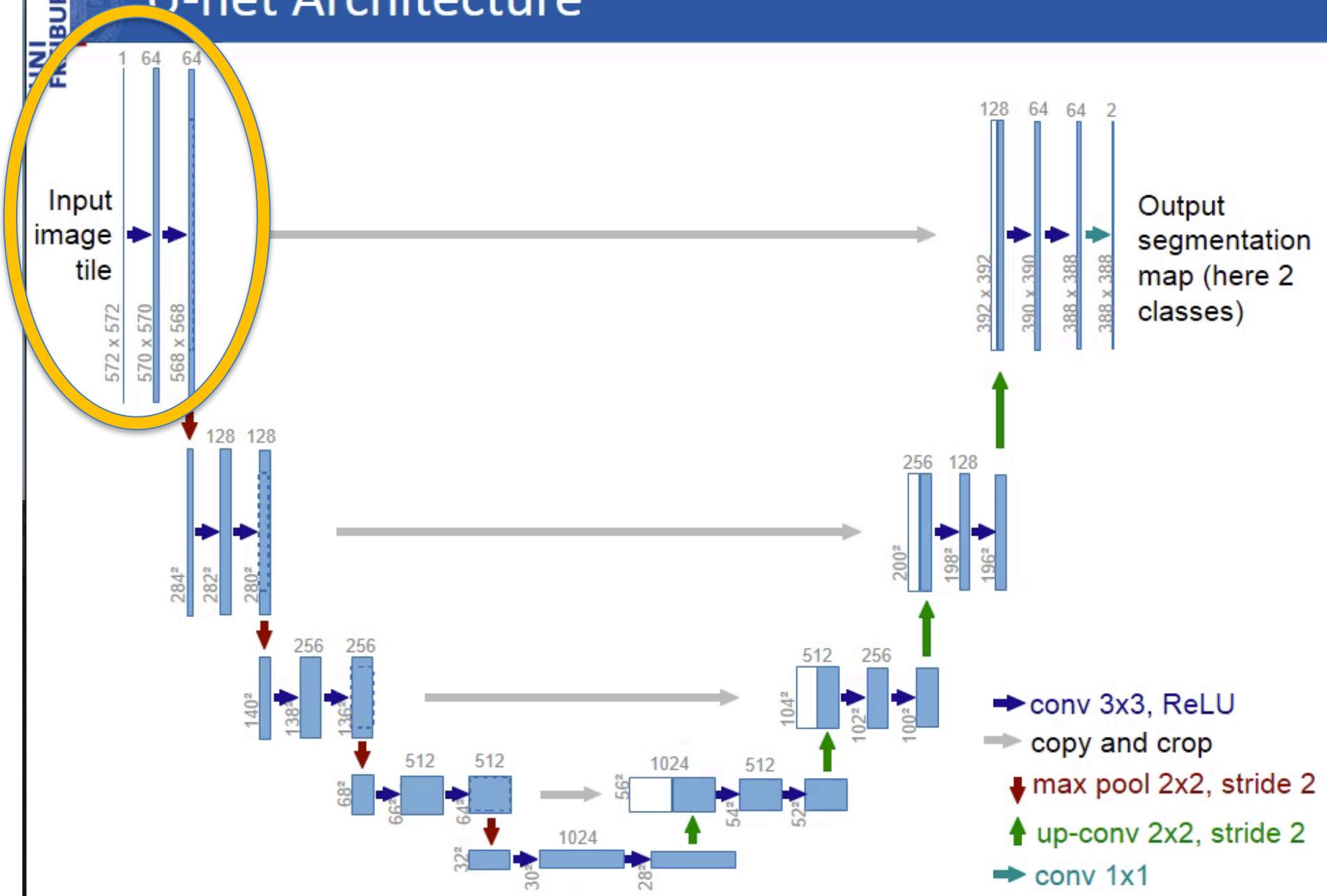


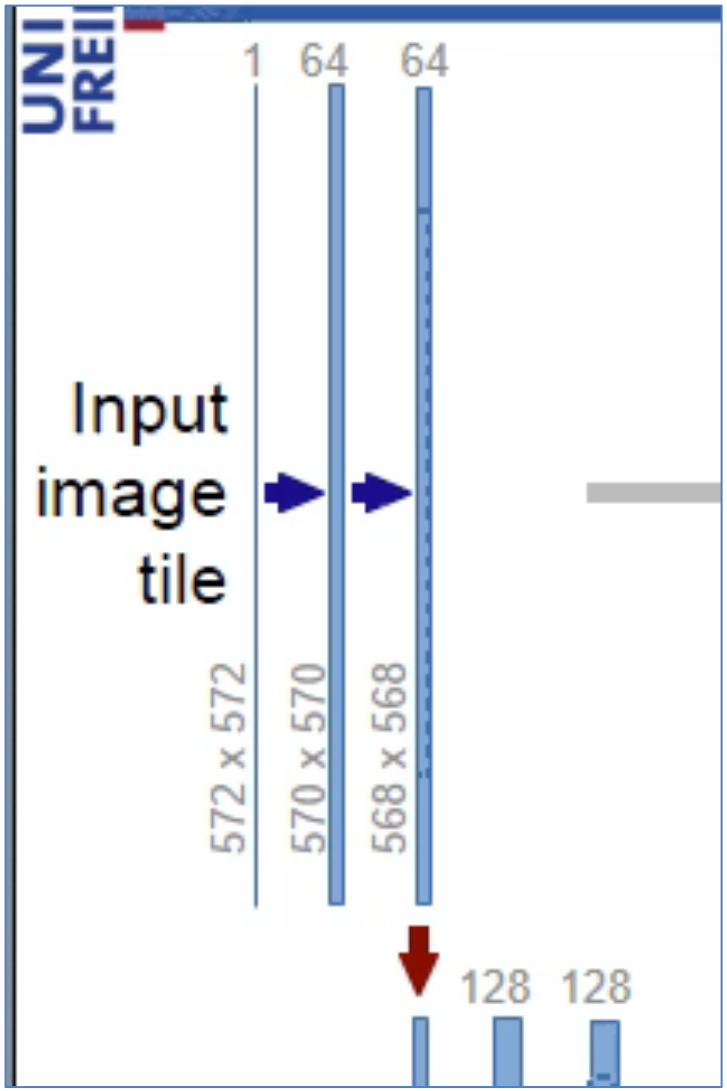
output segmentation map

- U-net **learns segmentation** in an end-to-end setting
- **Very few annotated images** (approx. 30 per application)
- **Touching objects** of the same class

[Data provided by Dr. Gert van Cappellen, Erasmus Medical Center. Rotterdam. The Netherlands]

U-net Architecture

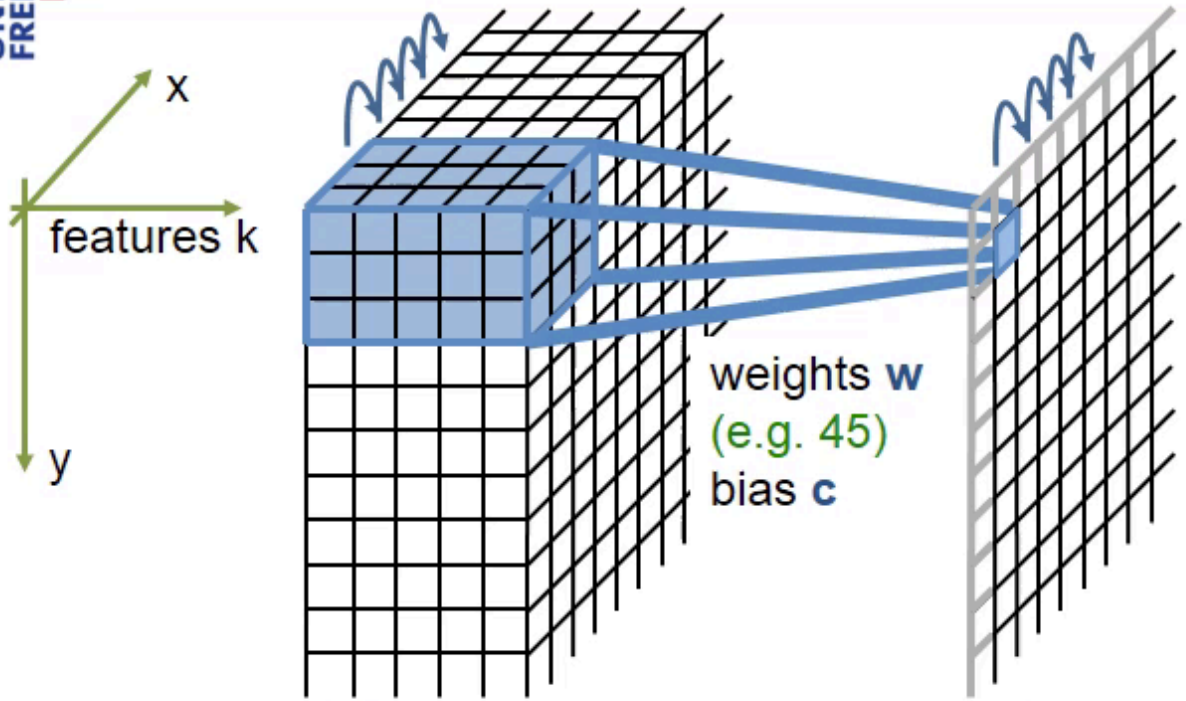




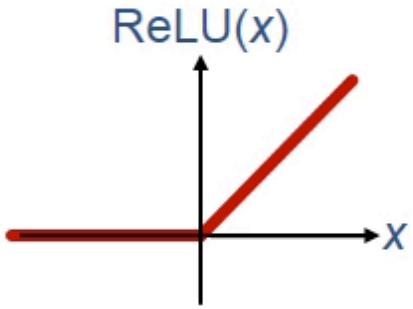
conv 3x3, ReLU
copy and crop
max pool 2x2, stride 2
up-conv 2x2, stride 2
conv 1x1

https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

3x3 convolution + ReLU

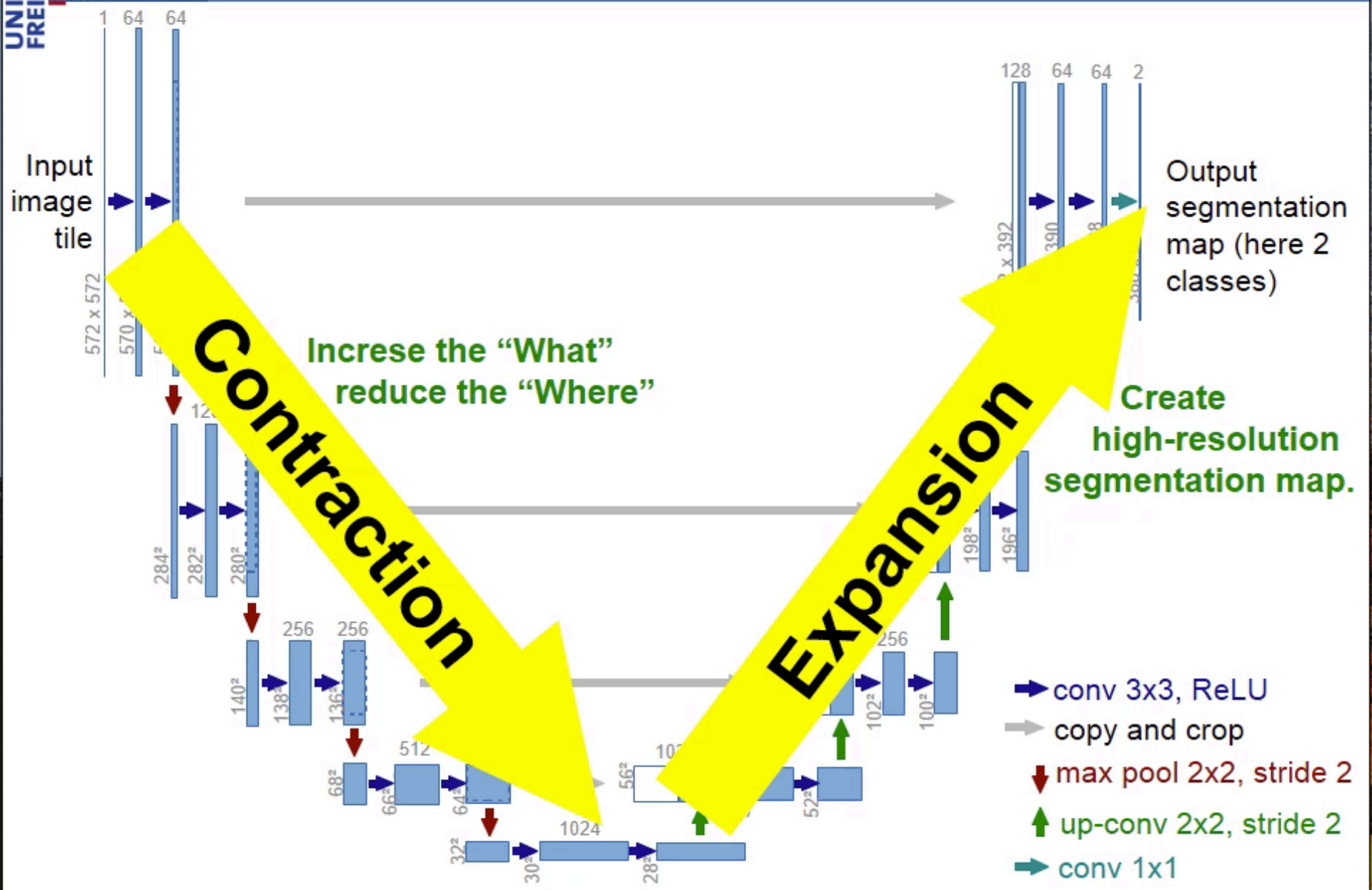


- Only valid part of convolution is used.
- For 3x3 convolutions a 1-pixel border is lost



$$b_{x,y,l} = \mathrm{ReLU}\Big(\sum_{\substack{i\in\{-1,0,1\}\\ j\in\{-1,0,1\}\\ k\in\{1,\ldots,K\}}} w_{i,j,k,l} \cdot a_{x+i,y+j,k} + c_l \Big)$$

U-Net Architecture



U-net Architecture

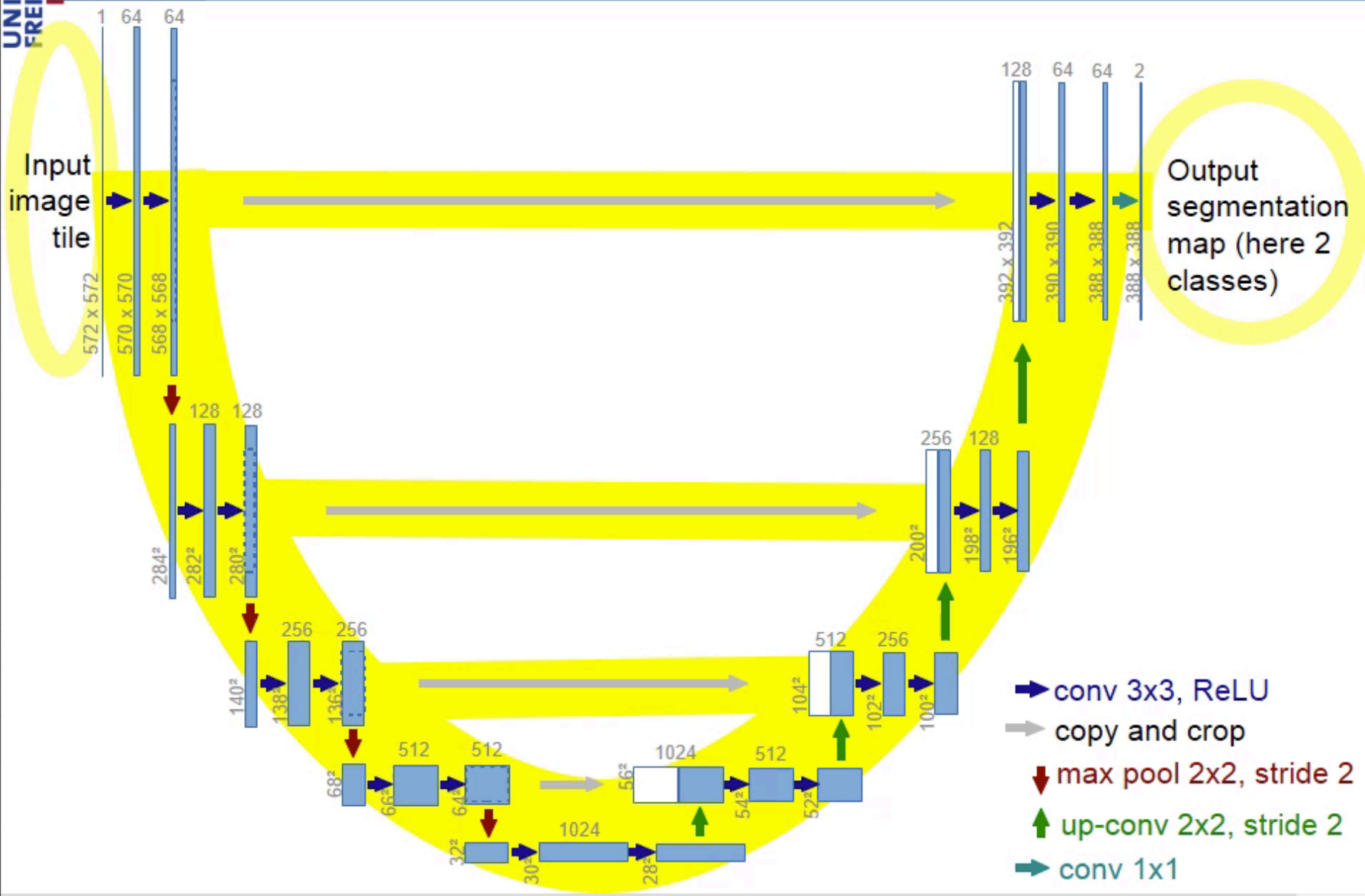
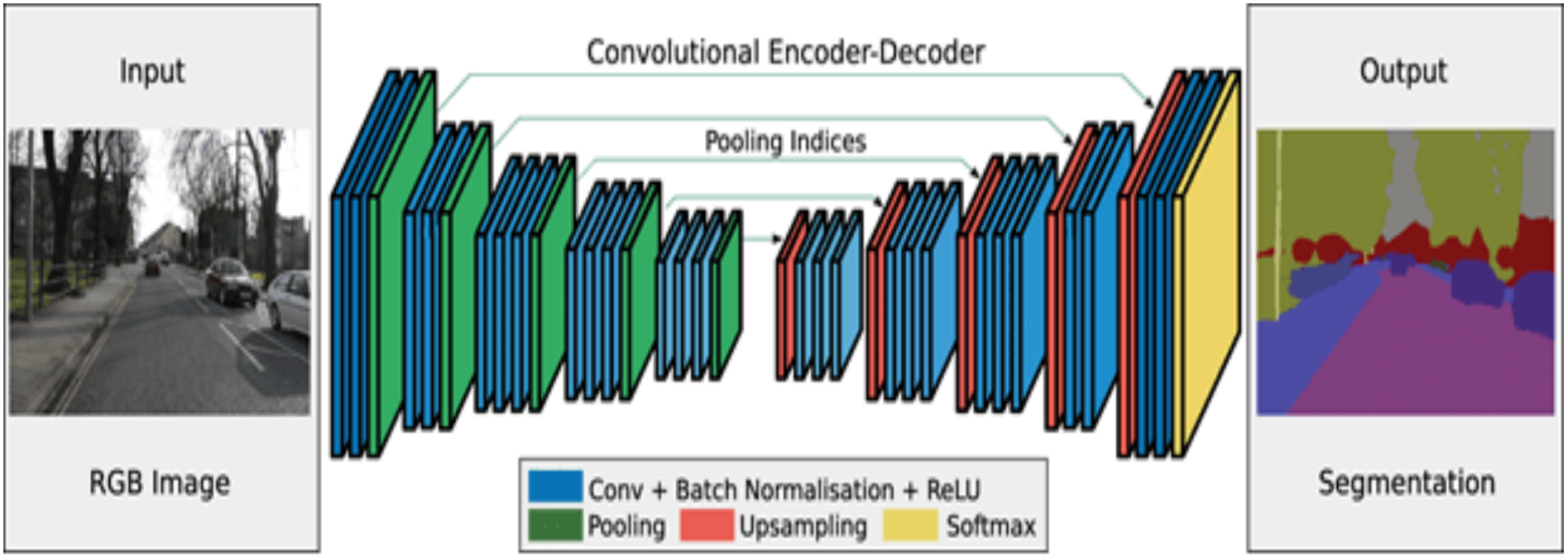


Image segmentation

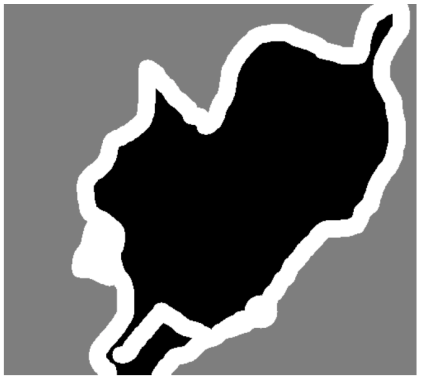
The basic architecture in image segmentation consists of an encoder and a decoder.



The encoder extracts features from the image through filters. The decoder is responsible for generating the final output which is usually a segmentation mask containing the outline of the object. Most of the architectures have this architecture or a variant of it.

https://neptune.ai/blog/image-segmentation-in-2020

CODE explanation
https://keras.io/examples/vision/oxford_pets_image_segmentation/



DNN Model

Applications

+

Data Ensemble + Input

+

It's all about

linear algebra calculations

DNN in particular

+

Algorithms, Software

+

Output + Data Analysis

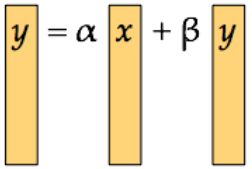
+

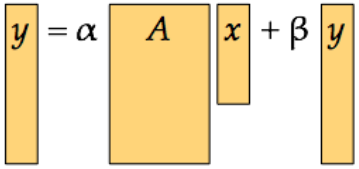
Hardware

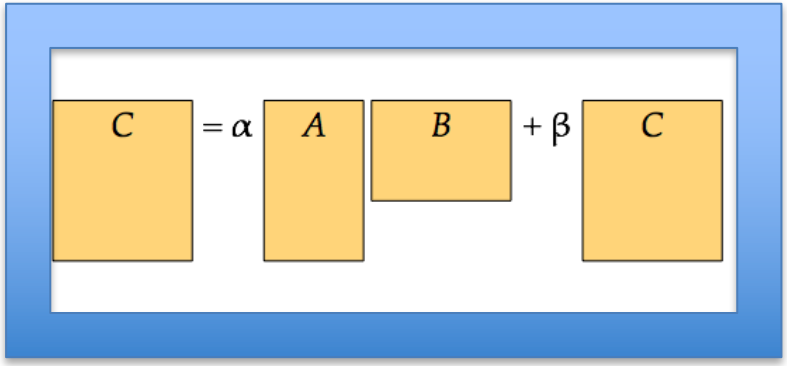
LOTS of MM (BLAS3), Parallel !!

Computational Intensity = FLOPS/ Memory Access

- ✓ Level 1 BLAS — vector operations
  - ✓ $O(n)$ data and flops (floating point operations)
  - ✓ Memory bound:
    $O(1)$ flops per memory access
- ✓ Level 2 BLAS — matrix-vector operations
  - ✓ $O(n^2)$ data and flops
  - ✓ Memory bound:
    $O(1)$ flops per memory access
- ✓ Level 3 BLAS — matrix-matrix operations
  - ✓ $O(n^2)$ data, $O(n^3)$ flops
  - ✓ Surface-to-volume effect
  - ✓ Compute bound:
    $O(n)$ flops per memory access







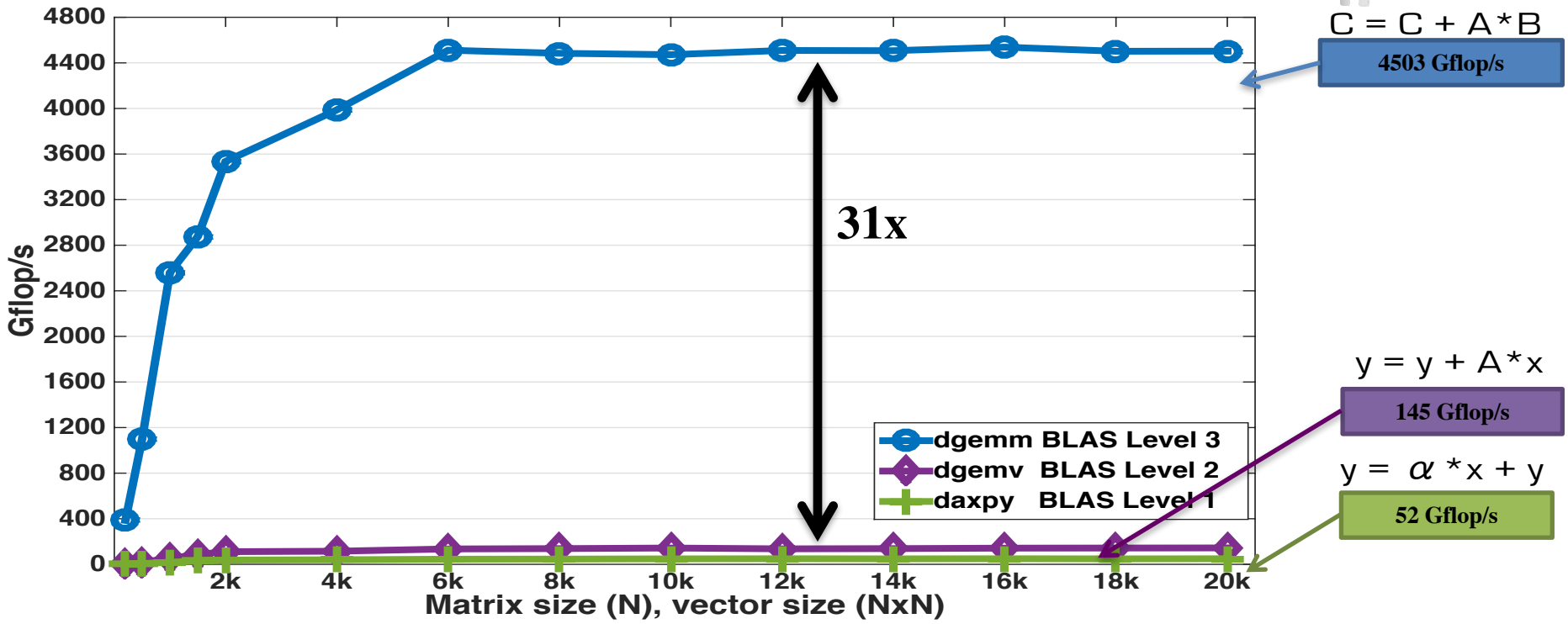
Level 1, 2 and 3 BLAS on GPU

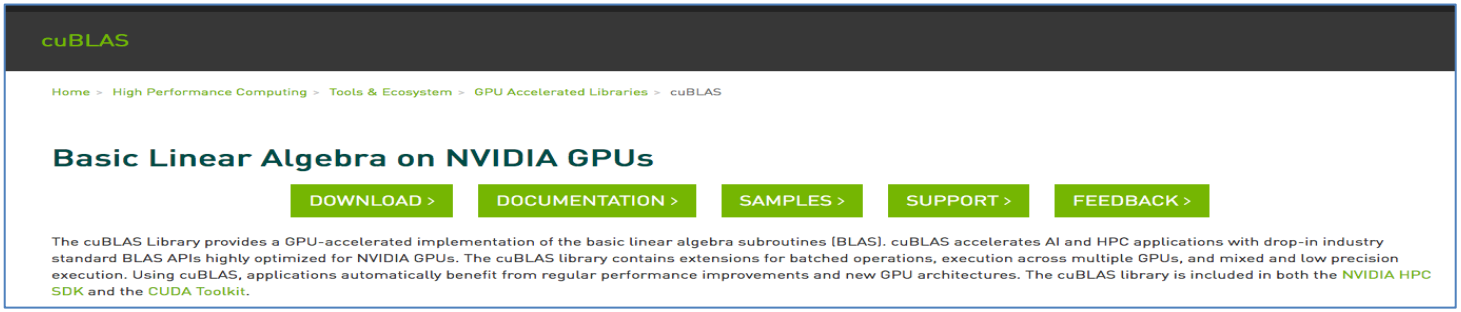
Joint Institute for Computational Sciences

INNOVATIVE COMPUTING LABORATORY

Nvidia P100, The theoretical peak double precision is 4700 Gflop/s, CUDA version 8.0

Nvidia **P100**, 1.19 GHz, Peak DP = 4700 Gflop/s





LAB 2
Problem 4

Write a python code to compute C=AXB and plot a curve of the FLOPS against the matrix size N (take N=2000, 4000, 6000) using single precision when the computation is done on a CPU. DO the same on the GPU

```
import numpy as np
import time

A = np.random.rand(2000, 2000).astype('float32')
B = np.random.rand(2000, 2000).astype('float32')
%timeit np.dot(A,B)
timestt = time.time()
C=np.dot(A,B)
timeend = time.time()
tmm2000 = timeend - timestt
gf2000 = 2*2*2*2/tmm2000
print('time = ',tmm2000)
print('GFLOPS = ', gf2000)

A = np.random.rand(4000, 4000).astype('float32')
B = np.random.rand(4000, 4000).astype('float32')
timestt = time.time()
C=np.dot(A,B)
timeend = time.time()
tmm4000 = timeend - timestt
gf4000 = 2*4*4*4/tmm4000
print('time = ',tmm4000)
print('GFLOPS = ', gf4000)
```

```
A = np.random.rand(6000, 6000).astype('float32')
B = np.random.rand(6000, 6000).astype('float32')
timestt = time.time()
C=np.dot(A,B)
timeend = time.time()
tmm6000 = timeend - timestt
gf6000 = 2*6*6*6/tmm6000
print('time = ',tmm6000)
print('GFLOPS = ', gf6000)

import torch
A = torch.randn(6000, 6000).cuda()
B = torch.randn(6000, 6000).cuda()
timestt = time.time()
C=torch.matmul(A,B)
timeend = time.time()
tmm6000 = timeend - timestt
gf6000 = 2*6*6*6/tmm6000
print('time = ',tmm6000)
print('GFLOPS = ', gf6000)
```

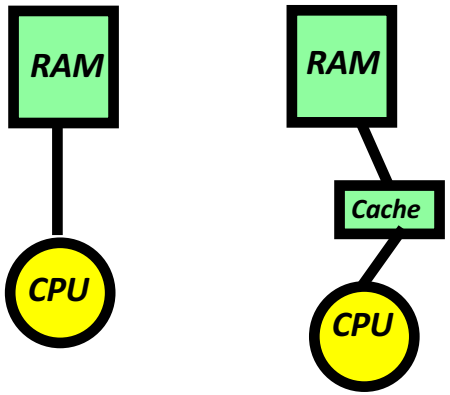
```
1 loop, best of 5: 228 ms per loop
time =  0.2360849380493164
GFLOPS =  67.7722184744277
time =  1.8578176498413086
GFLOPS =  68.8980428251037
time =  6.155170440673828
GFLOPS =  70.18489644824643
```

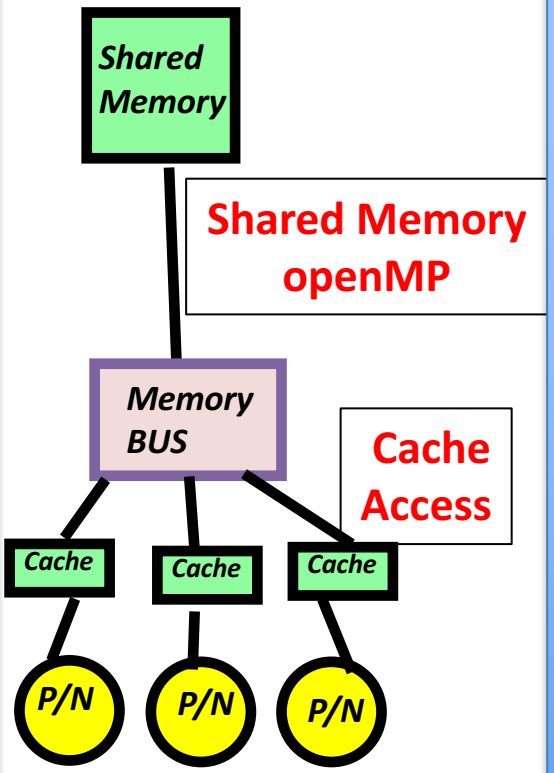
```
1 loop, best of 5: 214 ms per loop
time =  0.22870469093322754
GFLOPS =  69.95921218192831
time =  1.8288803100585938
GFLOPS =  69.98817762759944
time =  6.09820103645324 7
GFLOPS =  70.84056386754575
```

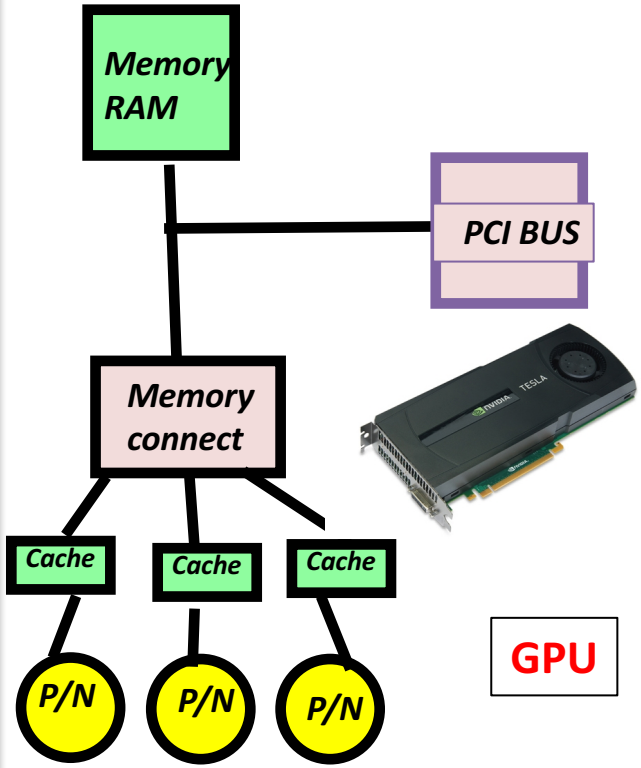
```
time =  6.029452800750732
GFLOPS =  71.6482928510878
time =  0.06104087829589844
GFLOPS =  7077.224510202169
```

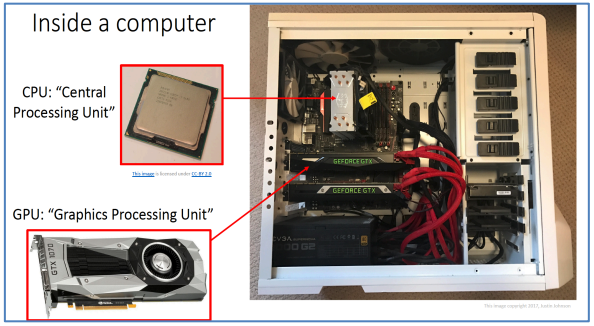
Simple story of Computers

Major Bottleneck - Communication, Memory Access









DNN Computing – Why GPU

Lots of MM (BLAS3) – Performance

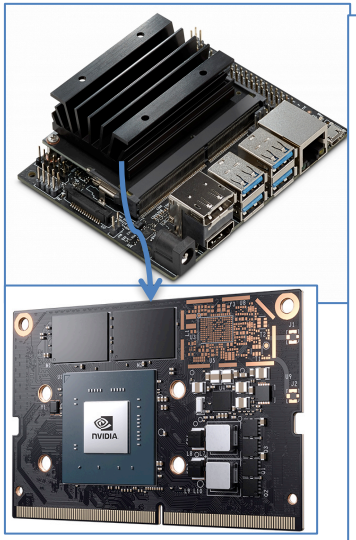
Data in 32 bits or 16 bits – Reduce Memory Access

Lots of Small Parallel Computing Unit – Time Scale

Power Usage - Low , Efficiency

Fast Memory Access Within GPU

GPU architectures



DEVELOPER KIT	
GPU	128-core Maxwell
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	microSD (not included)
Video Encoder	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)
Video Decoder	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30| (H.264/H.265)
Camera	1x MIPI CSI-2 DPHY lanes
Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI 2.0 and eDP 1.4
USB	4x USB 3.0, USB 2.0 Micro-B
Others	GPIO, I²C, I²S, SPI, UART
Mechanical	100 mm x 80 mm x 29 mm

Jetson Nano Developer Kit

Nsight Systems	2019.3
Nsight Graphics	2018.7
Nsight Compute	1.0
Jetson GPIO	1.0
Jetson OS	Ubuntu 18.04
CUDA	10.0.166
cuDNN	7.3.1.28
TensorRT	5.0.6.3

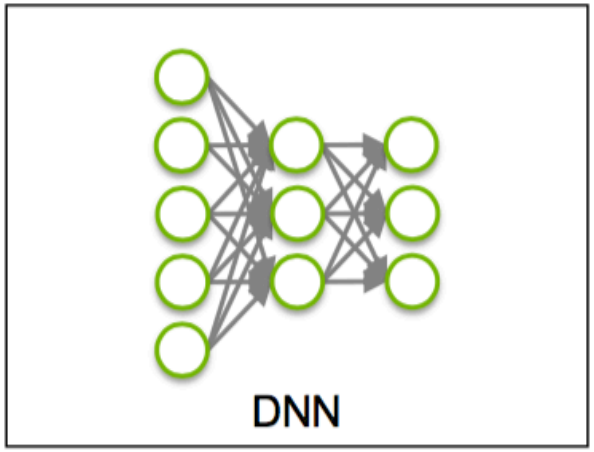
Other NVIDIA GPUs used in this workshop: GTX 1650 , K80, P100, V100, A100, ..

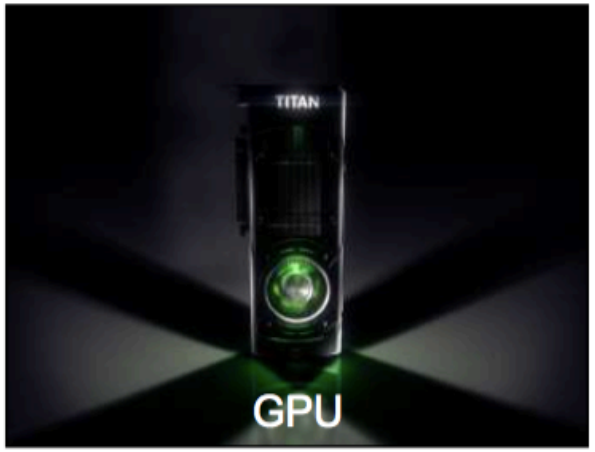
PRODUCT SPECIFICATIONS	
NVIDIA® CUDA Cores	896
Clock Speed	1485 MHz
Boost Speed	1725 MHz
Memory Speed (Gbps)	8
Memory Size	4GB GDDR5
Memory Interface	128-bit
Memory Bandwidth (Gbps)	128

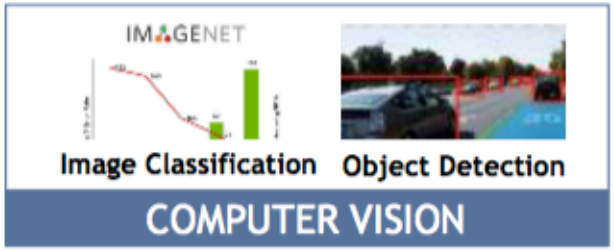
NVIDIA V100 on Summit

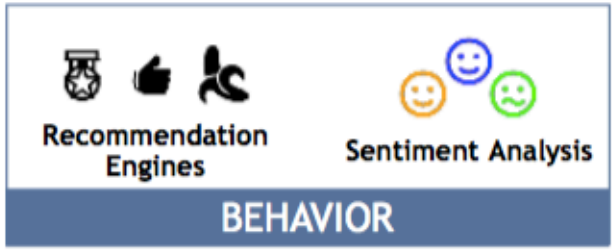
GPU Architecture	NVIDIA Volta	
NVIDIA Tensor Cores	640	
NVIDIA CUDA® Cores	5,120	
Double-Precision Performance	7 TFLOPS	7.8 TFLOPS
Single-Precision Performance	14 TFLOPS	15.7 TFLOPS
Tensor Performance	112 TFLOPS	125 TFLOPS
Memory Bandwidth	900GB/sec	

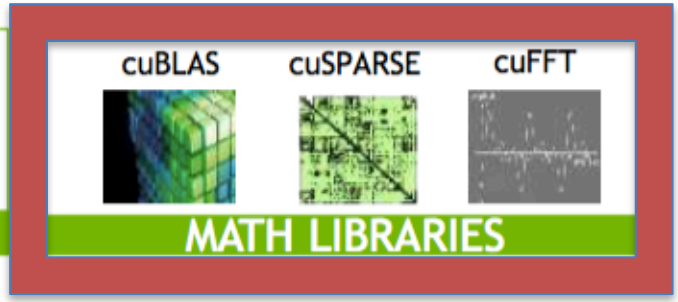
Machine Learning :LA (BLAS3) – GPU acceleration

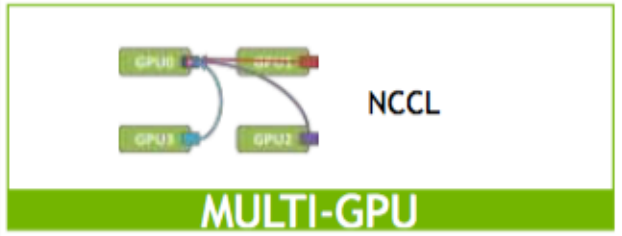












Source captured frin - Julie Bernauer – HPC Advisory Council Stanford Tutorial – 2017/02/07

GPU Performance architectures

GPU Features	NVIDIA Tesla P100	NVIDIA Tesla V100	NVIDIA A100
GPU Codename	GP100	GV100	GA100
GPU Architecture	NVIDIA Pascal	NVIDIA Volta	NVIDIA Ampere
GPU Board Form Factor	SXM	SXM2	SXM4
SMs	56	80	108
TPCs	28	40	54
FP32 Cores / SM	64	64	64
FP32 Cores / GPU	3584	5120	6912
FP64 Cores / SM (excl. Tensor)	32	32	32
FP64 Cores / GPU (excl. Tensor)	1792	2560	3456
INT32 Cores / SM	NA	64	64
INT32 Cores / GPU	NA	5120	6912
Tensor Cores / SM	NA	8	4[2]
Tensor Cores / GPU	NA	640	432
GPU Boost Clock	1480 MHz	1530 MHz	1410 MHz
Peak FP16 Tensor TFLOPS with FP16 Accumulate[1]	NA	125	312/624[3]
Peak FP16 Tensor TFLOPS with FP32 Accumulate[1]	NA	125	312/624[3]
Peak BF16 Tensor TFLOPS with FP32 Accumulate[1]	NA	NA	312/624[3]
Peak TF32 Tensor TFLOPS[1]	NA	NA	156/312[3]
Peak FP64 Tensor TFLOPS[1]	NA	NA	19.5
Peak INT8 Tensor TOPS[1]	NA	NA	624/1248[3]
Peak INT4 Tensor TOPS[1]	NA	NA	1248/2496[3]
Peak FP16 TFLOPS[1] (non-Tensor)	21.2	31.4	78
Peak BF16 TFLOPS[1] (non-Tensor)	NA	NA	39
Peak FP32 TFLOPS[1] (non-Tensor)	10.6	15.7	19.5
Peak FP64 TFLOPS[1] (non-Tensor)	5.3	7.8	9.7
Peak INT32 TOPS[1,4]	NA	15.7	19.5
Texture Units	224	320	432
Memory Interface	4096-bit HBM2	4096-bit HBM2	5120-bit HBM2
Memory Size	16 GB	32 GB / 16 GB	40 GB
Memory Data Rate	703 MHz DDR	877.5 MHz DDR	1215 MHz DDR

Peak FP64[1]	9.7 TFLOPS
Peak FP64 Tensor Core[1]	19.5 TFLOPS
Peak FP32[1]	19.5 TFLOPS
Peak FP16[1]	78 TFLOPS
Peak BF16[1]	39 TFLOPS
Peak TF32 Tensor Core[1]	156 TFLOPS | 312 TFLOPS[2]
Peak FP16 Tensor Core[1]	312 TFLOPS | 624 TFLOPS[2]
Peak BF16 Tensor Core[1]	312 TFLOPS | 624 TFLOPS[2]
Peak INT8 Tensor Core[1]	624 TOPS | 1,248 TOPS[2]
Peak INT4 Tensor Core[1]	1,248 TOPS | 2,496 TOPS[2]

Memory Bandwidth	720 GB/sec	900 GB/sec	1555 GB/sec
L2 Cache Size	4096 KB	6144 KB	40960 KB
Shared Memory Size / SM	64 KB	Configurable up to 96 KB	Configurable up to 164 KB
Register File Size / SM	256 KB	256 KB	256 KB
Register File Size / GPU	14336 KB	20480 KB	27648 KB
TDP	300 Watts	300 Watts	400 Watts
Transistors	15.3 billion	21.1 billion	54.2 billion
GPU Die Size	610 mm²	815 mm²	826 mm2
TSMC Manufacturing Process	16 nm FinFET+	12 nm FFN	7 nm N7

1. **Peak rates are based on GPU Boost Clock.**
2. **Four Tensor Cores in an A100 SM have 2x the raw FMA computational power of eight Tensor Cores in a GV100 SM.**
3. **Effective TOPS / TFLOPS using the new Sparsity Feature**
4. **TOPS = IMAD-based integer math**

Python programming

- ✓ Python is a programming language, (mycode.py)
- ✓ Many python programs for specific applications (libraries, packages) have been written by the community developers and could be “imported” to any python codes, e.g. numpy, pandas, etc..
- ✓ Jupyter notebook is a web-based interactive computing interface (GUI) mainly to run python (mycode.ipynb). It can be considered as a package in python.
- ✓ Anaconda is a software distribution managing a large collection of python packages primarily for data sciences applications. It is a popular platform to run python.
- ✓ Google Colab is web-based interface running jupyter notebook on computers provided by Google. It is a computing platform (free computers!!)

Colab → jupyter notebook → python → Google computers

Anaconda → jupyter notebook → python → your computer

Software – Operating Systems

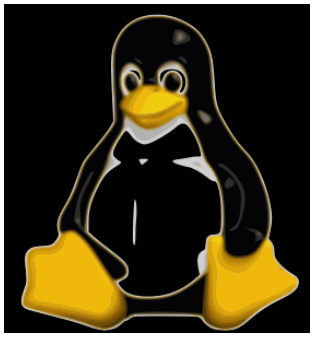
- ✓ Microsoft – Windows OS, 80% primarily for Desktop applications
- ✓ Mac – Mac OS – 10 % Desktop, BSD type, functionality similar to Linux
- ✓ Linux – open source, ubuntu, Centos, Redhat, Fedora, openSUSE, Debian,
- ✓ Entreprise : centos, Redat, ubuntu, SUSE, Cray OS..
- ✓ Phone, tablets : iOS, Android
- ✓ Virtual Box, VM,
- ✓ Container

- ✓ Cloud: Amazon, google cloud, MS Azure
- ✓ Free Access, Cheap device, …
- ✓ Web portal, google colab (GUI),
- ✓ Gateway, XSEDE gateway

Basics of LINUX OS

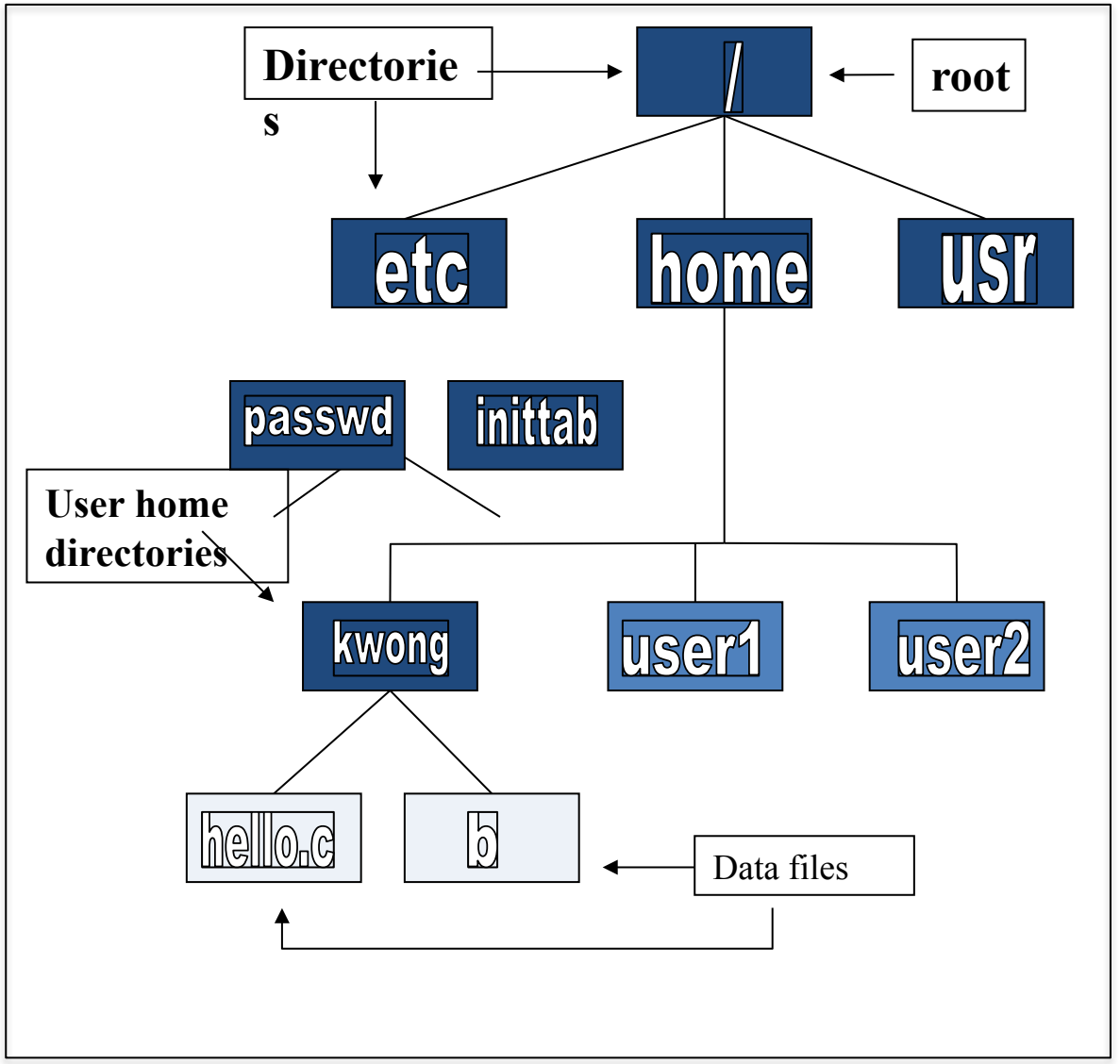
- ✓ Linux operation ? Skill level. Operating System, Software system, compiling
- ✓ In 70's, UNIX OS by Bell Lab, for main frame computer
- ✓ C is developed in 1972 by Dennis Ritchie at the Bell Laboratories for Unix OS
- ✓ In 80's, Microsoft's DOS, Apple MAC
- ✓ GNU project started by Richard Stallman in 1984 : free software, C compiler in 1991
- ✓ Linux Torvalds, a college sophomore, wrote the first Linux kernel in Sept. 1991 based on Minix developed by Anfrew Tanenbaum. UNIX on PC - LINUX
- ✓ www.linux .org, www.gnu.org

- ✓ Linux operating system in general
  - ✓ FILE, PATH, FILE MOD
- ✓ General overview Linux OS and terminal commands
  - ✓ file, program, executable program
  - ✓ cd , ls, mkdir, cp, mv , rm, xedit, gedit, env, path
- ✓ Tools and simple programming skills
- ✓ Compilers – gcc, g++
  - ✓ "gcc –o pexe ./prime.c" ; " ./pexe "



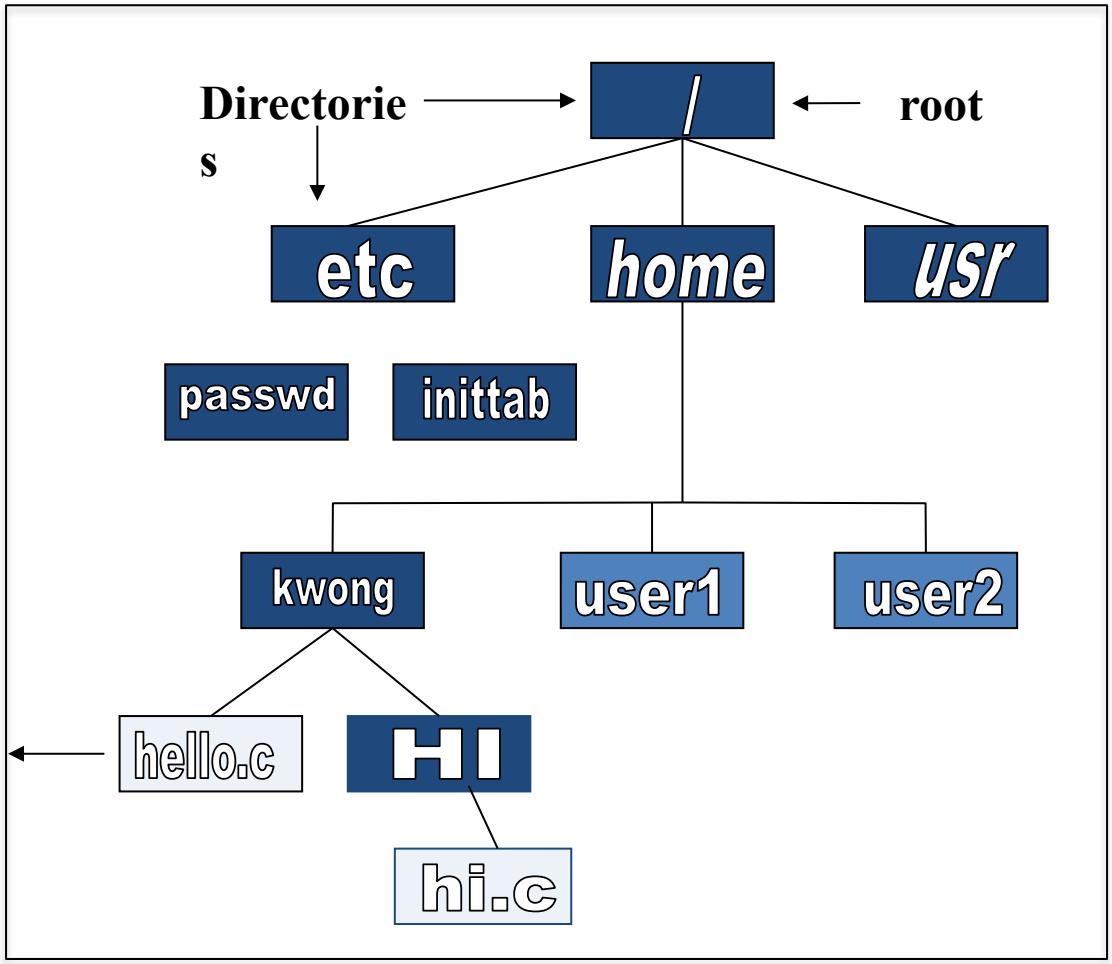
Linux File System

- Linux files are stored in a single rooted, hierarchical file system
  - Data files are stored in directories (folders)
  - Directories may be nested as deep as needed



Directory and Path : Absolute and Relative Path

```
$> cd
$>pwd
/home/kwong
$> ls
hello.c  HI
$>cd HI
$> ls
hi.c
$>pwd
/home/kwong/HI
$> cd ..
$> pwd
/home/kwong
```



- ✓ Absolute path - use pwd to find the address of the file, e.g. /home/kwong/CLASS/"file" in the CLASS directory"
- ✓ Relative path - use ./ to tell the computer the program is in the current directory, e.g use the ./"selected file in the current directory"

File Permissions : read (r), write (w), execute(x)

✓ The long version of a file listing (`ls -l`) will display the file permissions:

```
-rwxrwxr-x   1 kwong   kwong       5224 Dec 30 03:22 hexe
-rw-rw-r--   1 kwong   kwong        221 Dec 30 03:59 hello.c
-rw-rw-r--   1 kwong   kwong       1514 Dec 30 03:59 hello.s
drwxrwxr-x   7 kwong   kwong       1024 Dec 31 14:52 HI
```

Permissions Owner Group

-rwx rwx rwx (777) READ ~ 4 ; WRITE ~ 2 ; EXECUTE ~ 1 === total ~ 7

Other permissions
Group permissions
Owner permissions
Directory flag (d=directory; l=link)

```
chmod 755 file  # Owner=rwx Group=r-x Other=r-x
chmod 500 file2 # Owner=r-x Group=--- Other=---

chmod 644 file3 # Owner=rw- Group=r-- Other=r--
chmod +x  file  # Add execute permission to file for all
chmod o-r file  # Remove read permission for others
chmod a+w file  # Add write permission for everyone
```

Linux Commands

http://bootlin.com/doc/legacy/command-line/

LINUX COMMANDS CHEAT SHEET

SYSTEM

```
#uname -a          =>Displaylinux system information
#uname -r          =>isplay kernel release information
#uptime            =>Show how long the system has been running + load
#hostname          =>Show system host name
#hostname -i       =>Display the IP address of the host
#last reboot       =>Show system reboot history
#date              =>Show the current date and time
#cal               =>Show this month calendar
#w                 =>Display who is online
#whoami            =>Who you are logged in as
#finger user       =>Display information about user
```

HARDWARE

```
#dmesg             =>Detected hardware and boot messages
#cat /proc/cpuinfo =>CPU model
#cat /proc/meminfo =>Hardware memory
#cat /proc/interrupts =>Lists the number of interrupts per CPU per I/O device
#lshw              =>Displays information on hardware configuration of the
                     system
#lsblk             =>Displays block device related information in Linux
#free -m           =>Used and free memory (-m for MB)
#lspci -tv         =>Show PCI devices
#lsusb -tv         =>Show USB devices
#dmidecode         =>Show hardware info from the BIOS
#hdparm -i /dev/sda      =>Show info about disk sda
#hdparm -tT /dev/sda     =>Do a read speed test on disk sda
#badblocks -s /dev/sda   =>Test for unreadable blocks on disk sda
```

USERS

```
#id                =>Show the active user id with login and group
#last              =>Show last logins on the system
#who               =>Show who is logged on the system
#groupadd admin    =>Add group "admin"
#useradd -c "Sam Tomshi"   =>g admin -m sam #Create user "sam"
#userdel sam       =>Delete user sam
#adduser sam       =>Add user "sam"
#usermod           =>Modify user information
```

FILE COMMANDS

```
#ls –al            =>Display all information about files/ directories
#pwd               =>Show the path of current directory
#mkdir directory-name  =>Create a directory
#rm file-name          =>Delete file
#rm -r directory-nam   =>Delete directory recursively
#rm -f file-name       =>Forcefully remove file
#rm -rf directory-name =>Forcefully remove directory recursively
#cp file1 file2    =>Copy file1 to file2
#cp -r dir1 dir2   =>Copy dir1 to dir2, create dir2 if it doesn't exist
#mv file1 file2    =>Rename source to dest / move source to directory
```

cont.

```
#ln –s /path/to/file-name link-name   #Create symbolic link to file-name
#touch file        =>Create or update file
#cat > file        =>Place standard input into file
#more file         =>Output contents of file
#head file         =>Output first 10 lines of file
#tail file         =>Output last 10 lines of file
#tail -f file      =>Output contents of file as it grows starting with the
                     last 10 lines
#gpg -c file       =>Encrypt file
#gpg file.gpg      =>Decrypt file
#wc                =>print the number of bytes, words, and lines in files
#xargs             =>Execute command lines from standard input
```

PROCESS RELATED

```
#ps                =>Display your currently active processes
#ps aux | grep 'telnet'  =>Find all process id related to telnet process
#pmap              =>Memory map of process
#top               =>Display all running processes
#killpid           =>Kill process with mentioned pid id
#killall proc      =>Kill all processes named proc
#pkill process-name =>Send signal to a process with its name
#bg                =>[stopped] or [background] job
#fg                =>Bring the most recent job to foreground
#fg n              =>Bring job n to the foreground
```

FILE PERMISSION RELATED

```
#chmod octal file-name   =>Change the permissions of file to octal
Example
#chmod 777 /data/test.c  =>Set rwx permission for owner,group,world
#chmod 755 /data/test.c  =>Set rwx permission for owner,rw for group
                           and world
#chown owner-user file   =>Change owner of the file
#chown owner-user:owner-group file-name   =>Change owner and group
                                            owner of the file
#chown owner-user:owner-group directory   =>Change owner and group
                                            owner of the directory
```

NETWORK

```
#ifconfig –a       =>Display all network ports and ip address
#ifconfig eth0     =>Display specific ethernet port
#ethtool eth0      =>Linux tool to show ethernet status
#mii-tool eth0     =>Linux tool to show ethernet status
#ping host         =>Send echo request to test connection
#whois domain      =>Get who is information for domain
#dig domain        =>Get DNS information for domain
#dig -x host       =>Reverse lookup host
#host google.com   =>Lookup DNS ip address for the name
#hostname –i       =>Lookup local ip address
#wget file         =>Download file
#netstat -tupl     =>List active connections to / from system
```

COMPRESSION / ARCHIVES

```
#tar cf home.tar home    =>Create tar named home.tar containing home/
#tar xf file.tar         =>Extract the files from file.tar
#tar czf file.tar.gz files  =>Create a tar with gzip compression
#gzip file               =>Compress file and renames it to file.gz
```

INSTALL PACKAGE

```
#rpm -i pkgname.rpm   =>Install rpm based package
#rpm -e pkgname       =>Remove package
```

INSTALL FROM SOURCE

```
#./configure
#make
#make install
```

SEARCH

```
#grep pattern files   =>Search for pattern in files
#grep -r pattern dir  =>Search recursively for pattern in dir
#locate file          =>Find all instances of file
#[illegible]          =>Find files names that start with "index"
#[illegible]          =>Find files larger than 10000k in /home
```

LOGIN (SSH AND TELNET)

```
#ssh user@host        =>Connect to host as user
#ssh -p port user@host  =>Connect to host using specific port
#telnet host          =>Connect to the system using telnet port
```

FILE TRANSFER

```
scp
#scp file.txt server2:/tmp   =>Secure copy file.txt to remote host
                               /tmp folder
rsync
#rsync -a /home/apps /backup/   =>Synchronize source to destination
```

DISK USAGE

```
#df –h             =>Show free space on mounted filesystems
#df -i             =>Show free inodes on mounted filesystems
#fdisk -l          =>Show disks partitions sizes and types
#du -ah            =>Display disk usage in human readable form
#du -sh            =>Display total disk usage on the current directory
```

DIRECTORY TRAVERSE

```
#cd ..             =>To go up one level of the directory tree
#cd                =>Go to $HOME directory
#cd /test          =>Change to /test directory
```

MORE DETAILED : HTTP://LINOXIDE.COM/GUIDE/LINUX-COMMAND-SHELF.HTML

Linux Cheat Sheet

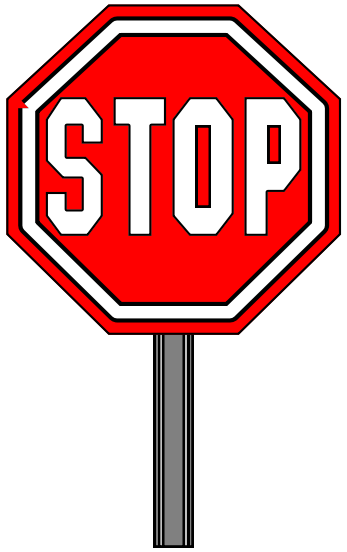
Keynote Movie

www.nvidia.com/gtc/keynote

The shared drive containing all the files including the video:

https://drive.google.com/drive/folders/0ABHc0USz4a5RUk9PVA

The End



- The End!