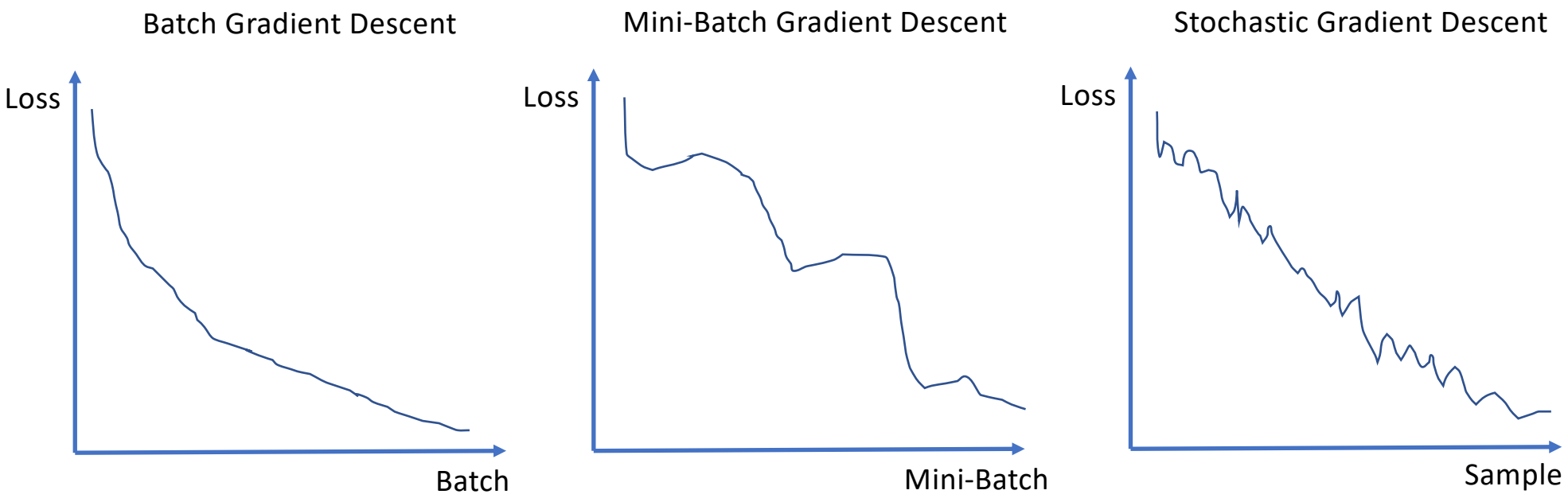
Convolutional Neural Networks (CNN)

John Tan Chong Min

What to focus on

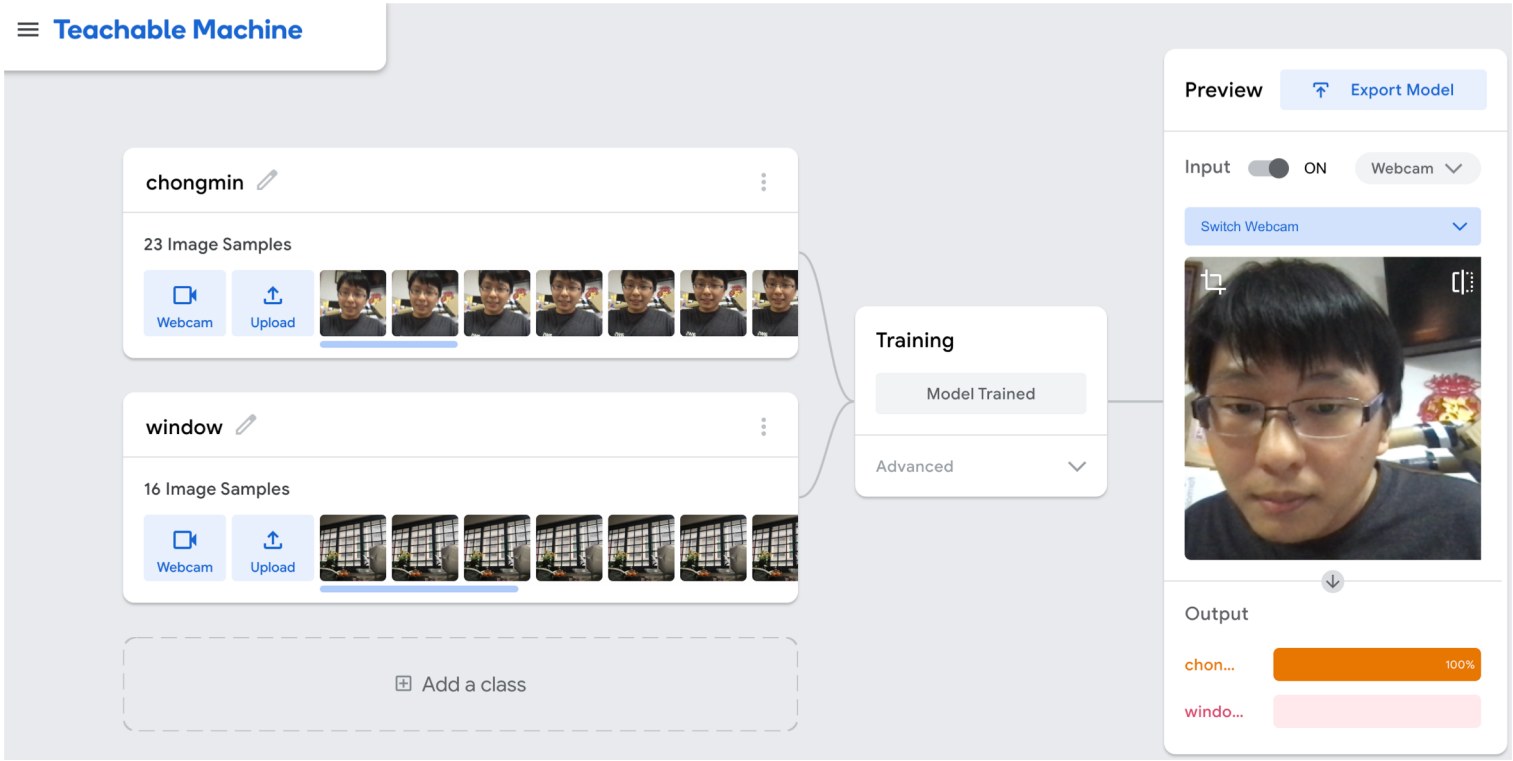
- Focus more on the concepts, less on the math
- No need to worry about the math equations
- Ask questions anytime: All questions are good. Questions help with understanding
- Every week: Theory first, then coding practical

Recap: Batch vs Stochastic Gradient Descent



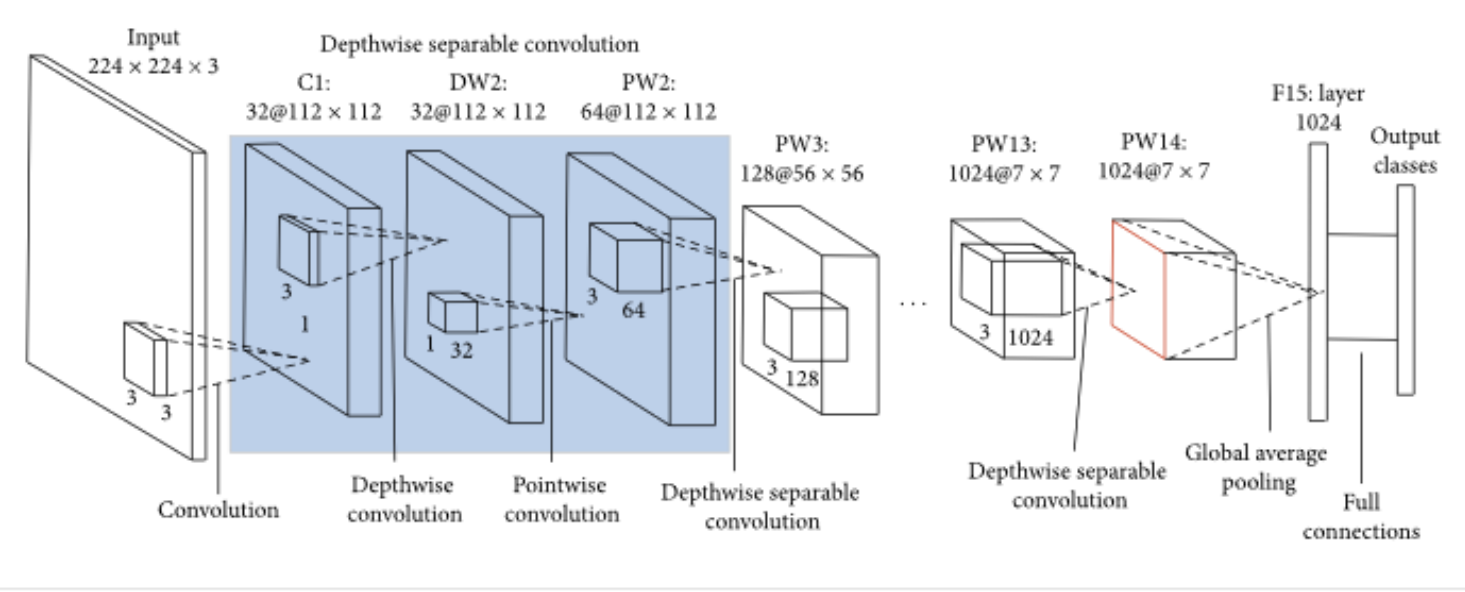
Try it out!

- https://teachablemachine.withgoogle.com/train/image



Under The Hood

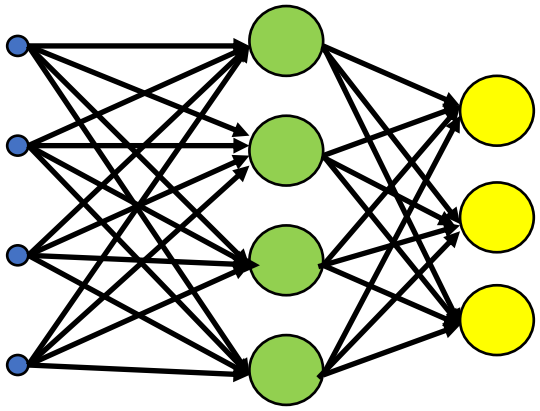
- Uses a type of Convolutional Neural Network (CNN)
- MobileNet



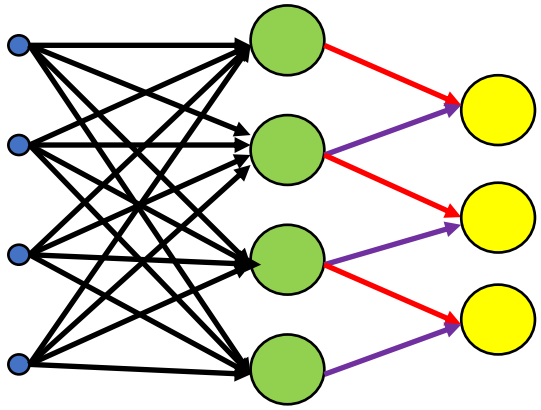
https://medium.com/analytics-vidhya/image-classification-with-mobilenet-cc6fbb2cd470

Why CNNs?

- What if we can reuse the same structure in multiple parts of the network?
- What if we only need to have connections within a local neighbourhood?
- Would it be better to just train that same structure once, and use it in many areas?
- Introducing…
  - Filters & Convolution!

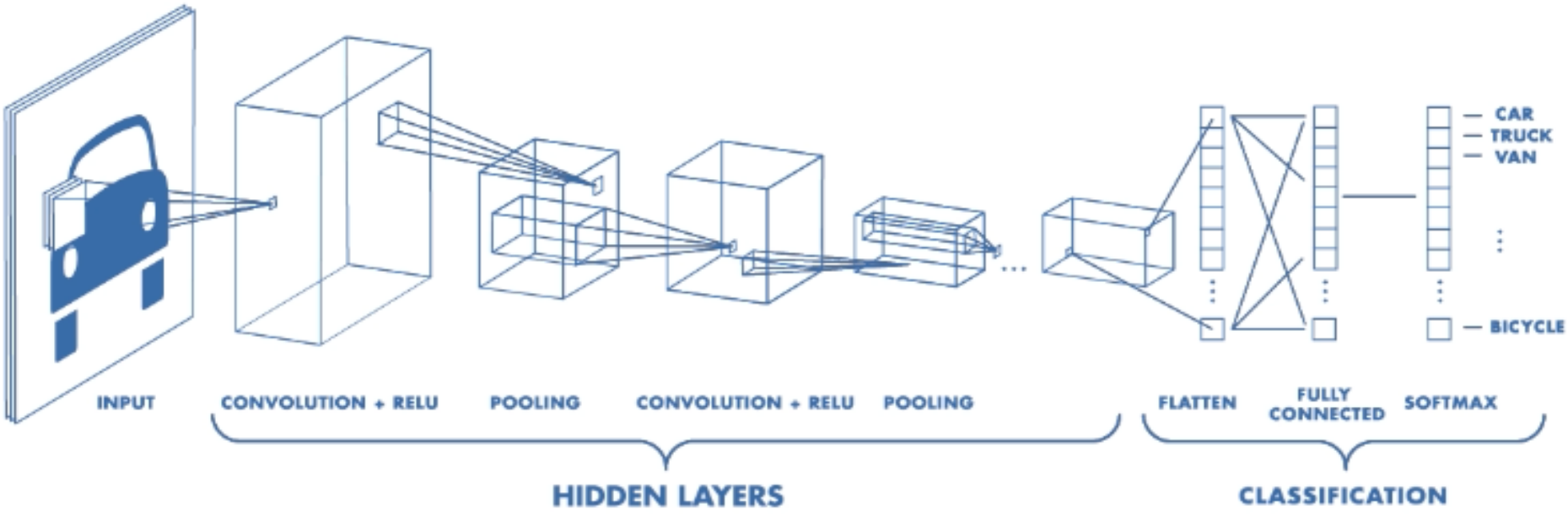


Fully-Connected (MLP)

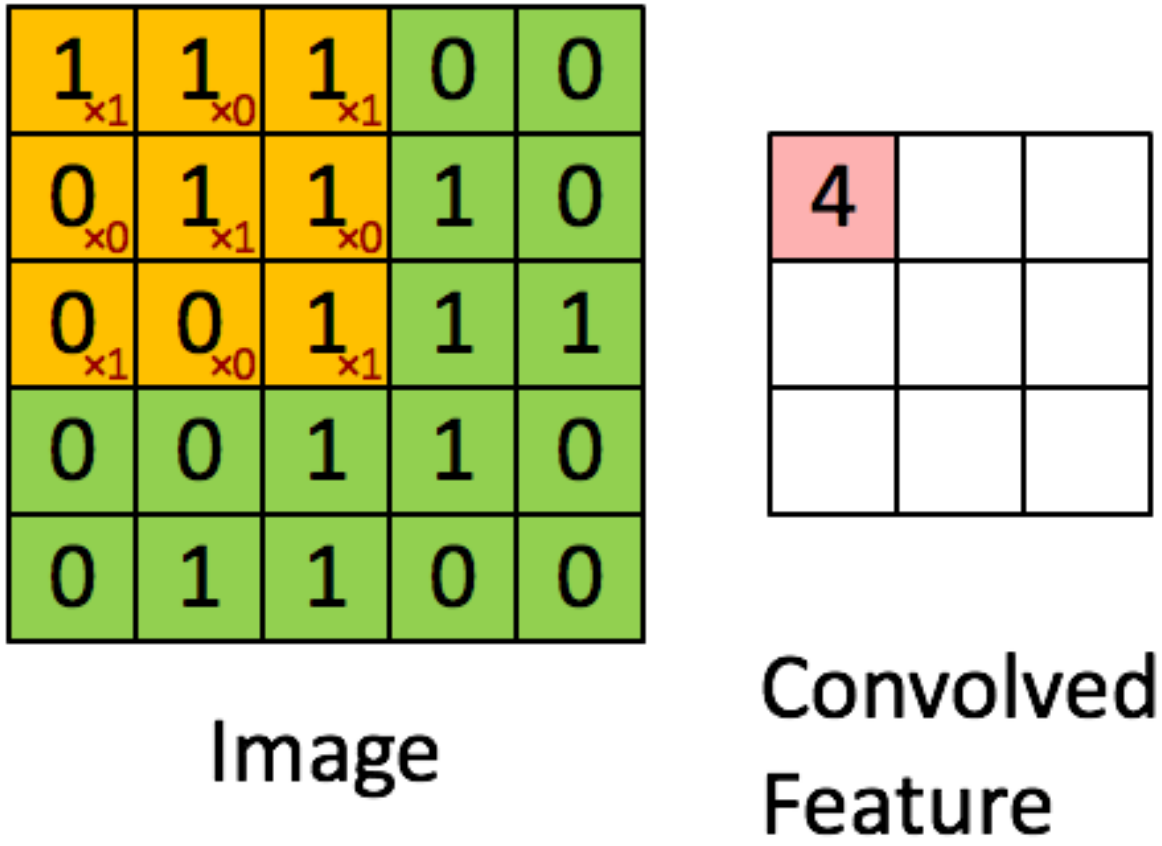


CNN-like (Sparse, weight sharing)

Typical Architecture of CNN

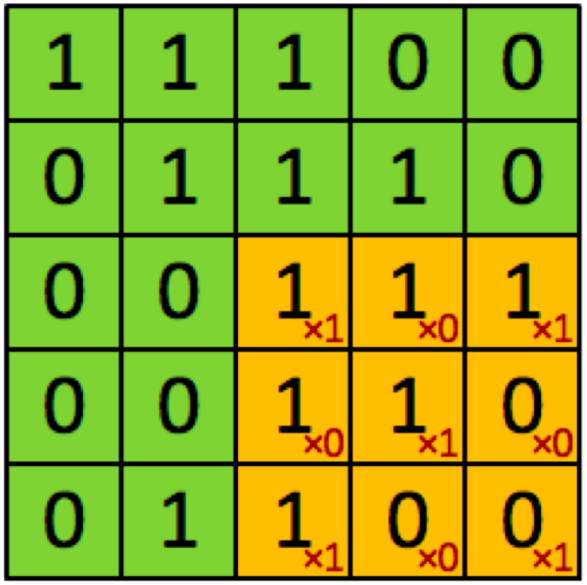


Convolution with Kernel/Filter

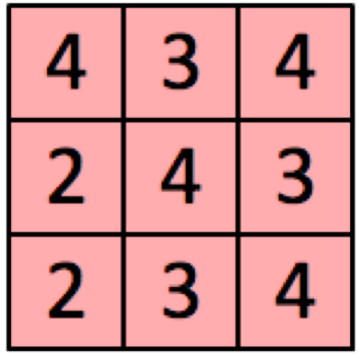


Convolution with Kernel/Filter

- Shared weights via convolution with filter across positions of an image
- Filter weights encode a feature



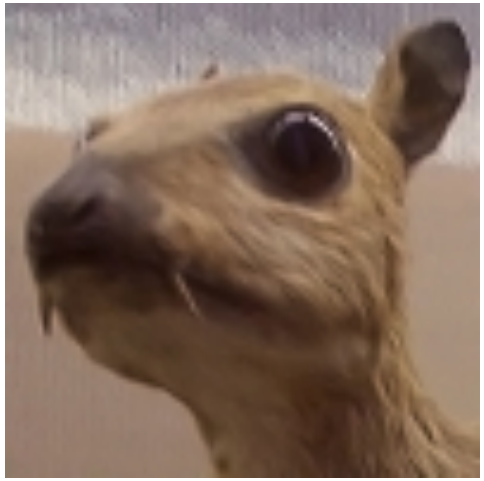
Image



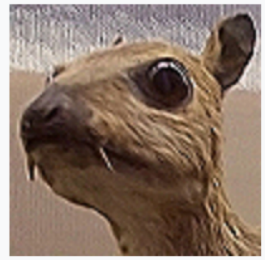
Convolved Feature

Traditional Kernels/Filters

- Handcrafted



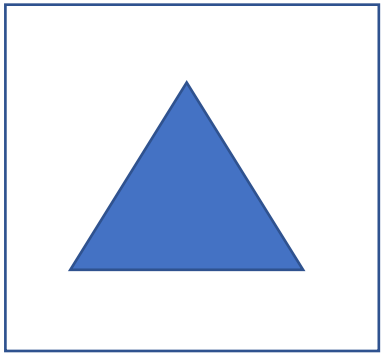
Original Image

Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

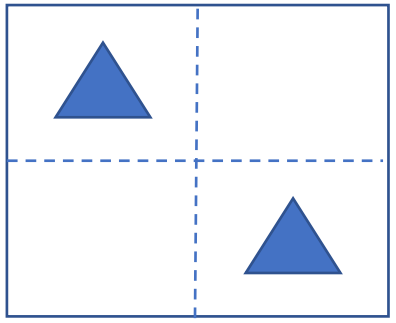
https://en.wikipedia.org/wiki/Kernel_(image_processing)

CNN Kernel/Filters

- No need for handcrafted kernels (kernel weights learnt by backpropagation)
- Has additional bias term and typically ReLU activation applied to output
  - Can allow distinct features to be indicated as only values past a certain threshold determined by the bias term will have non-zero output
- Can think of kernel/filters as providing a kind of template to identify certain features



Kernel



Image

1	0
0	1

Result after convolution

Padding and Strides

- Padding:
  - Extension of the boundary
- Stride:
  - How much to “shift” the kernel by

No padding
Strides: (1,1)

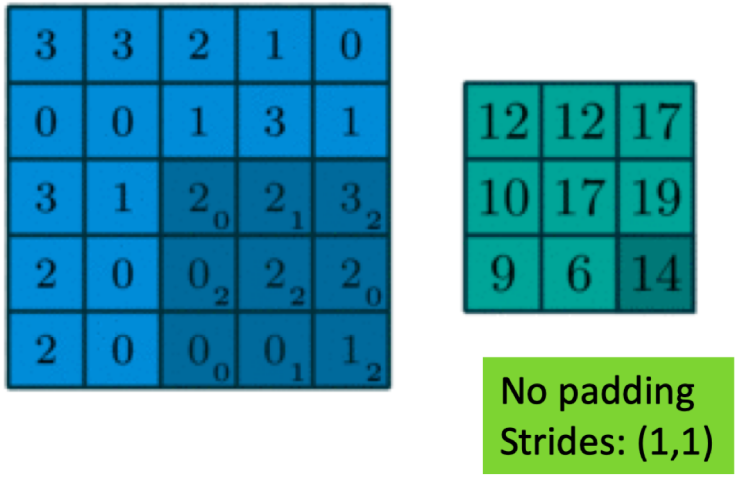
Padding: (1,1) zero padding
Strides: (2,2)

Padding and Strides

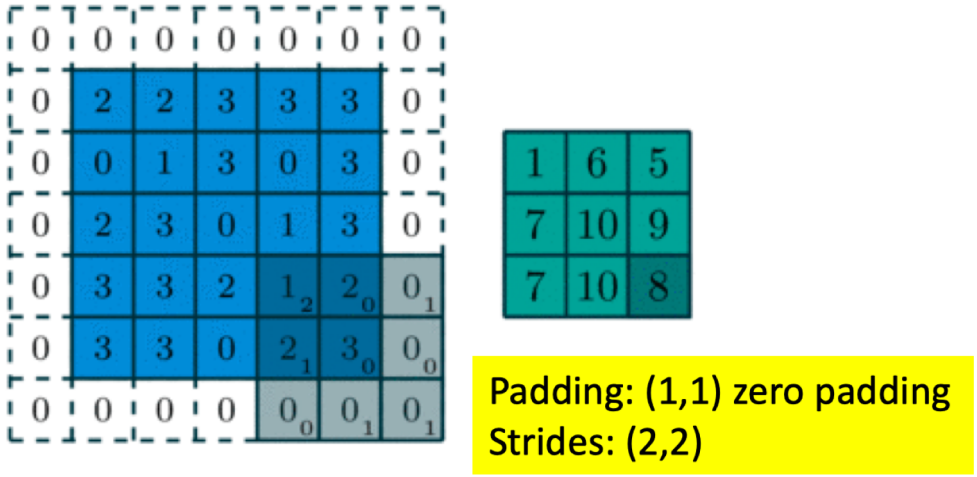
- Padding:
  - Extension of the boundary
- Stride:
  - How much to “shift” the kernel by

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

n_{in}: number of input features
n_{out}: number of output features
k: convolution kernel size
p: convolution padding size
s: convolution stride size



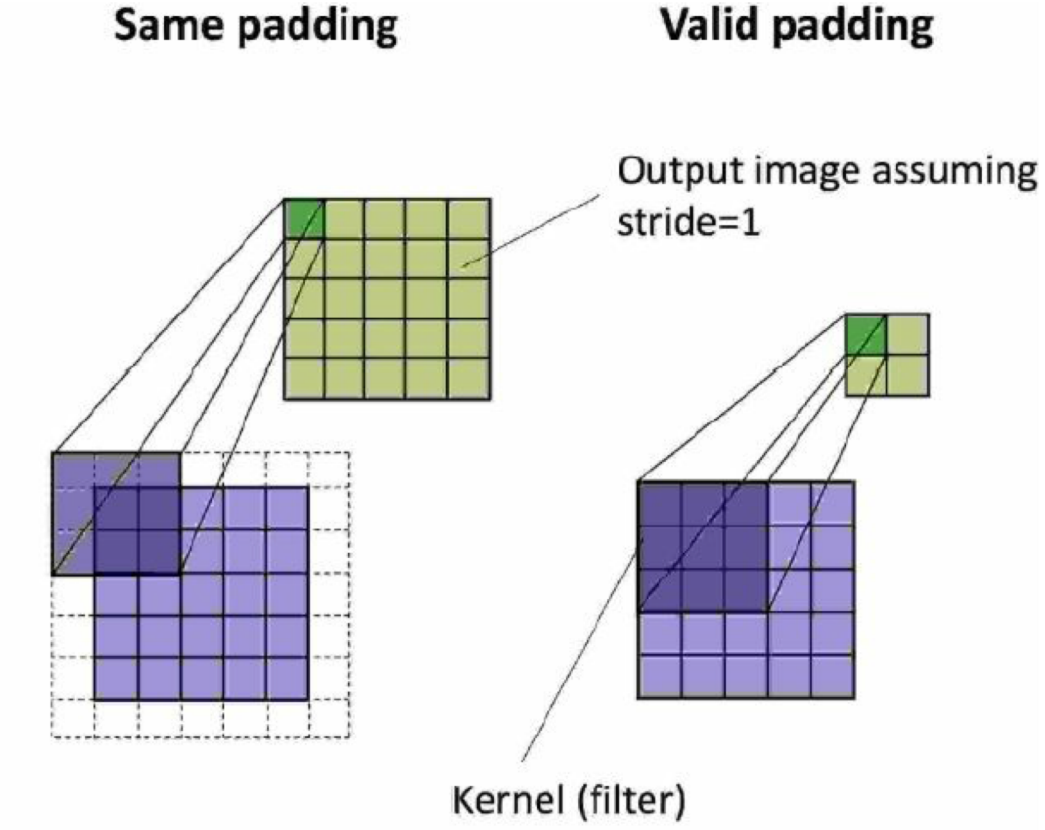
No padding
Strides: (1,1)



Padding: (1,1) zero padding
Strides: (2,2)

Types of Padding

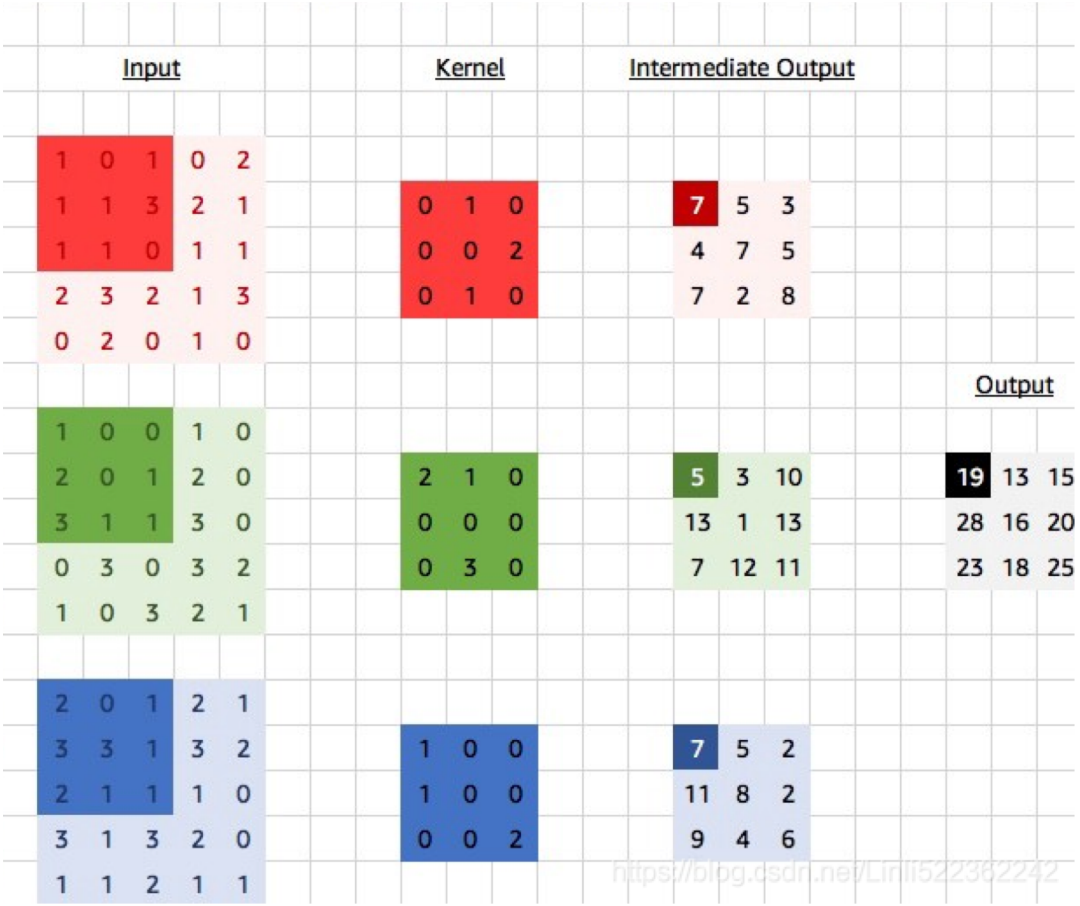
- Same:
  - Preserves input dimensions by padding accordingly
- Valid:
  - No padding at all



https://blog.csdn.net/Linli522362242/article/details/108414534

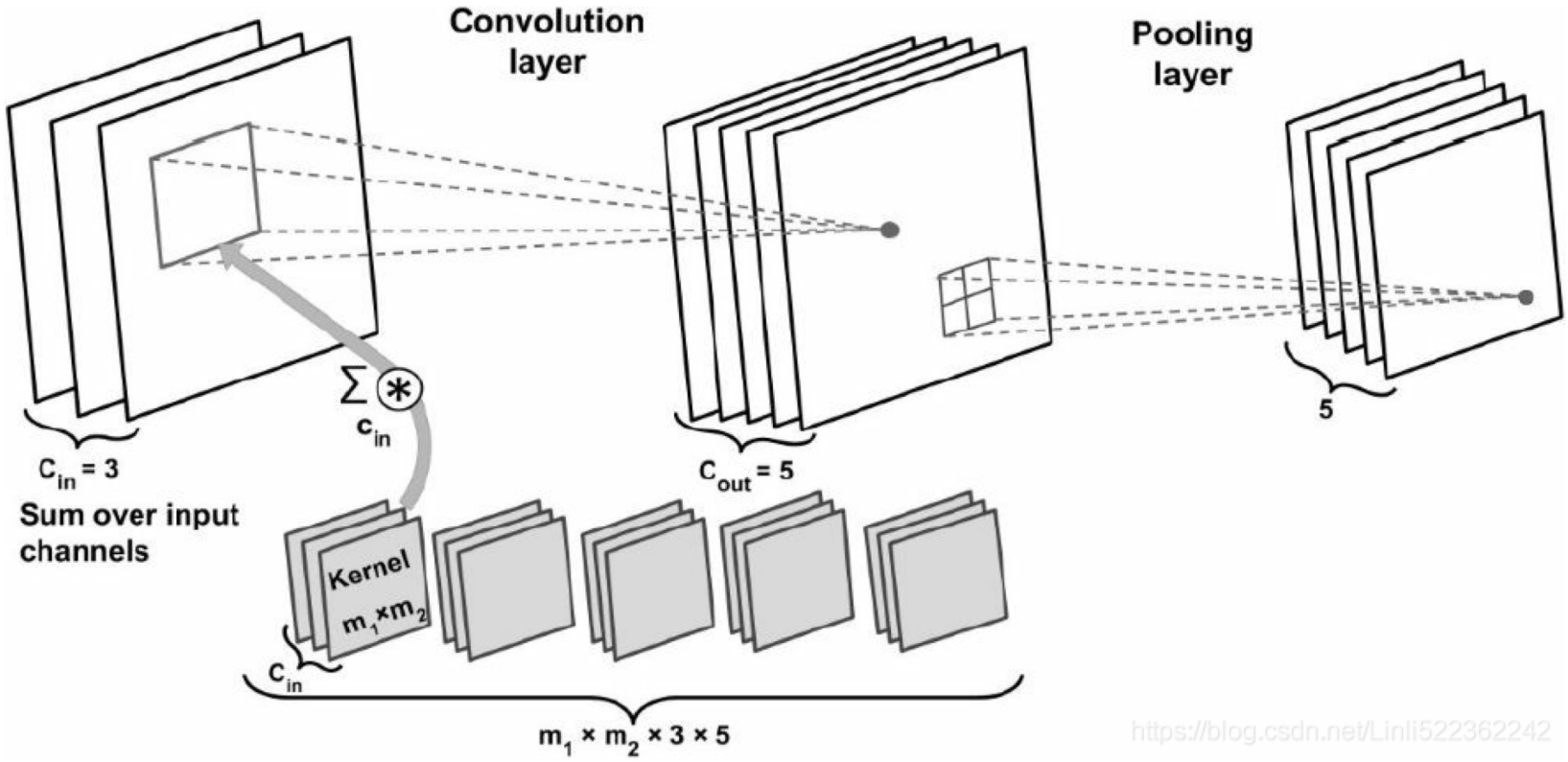
Multi-channel convolution

- Perform convolution for each input channel separately (can have different kernel weights for each channel)
- Then sum up all the outputs from the convolution for each channel



https://blog.csdn.net/Linli522362242/article/details/108414534

Multi-channel convolution



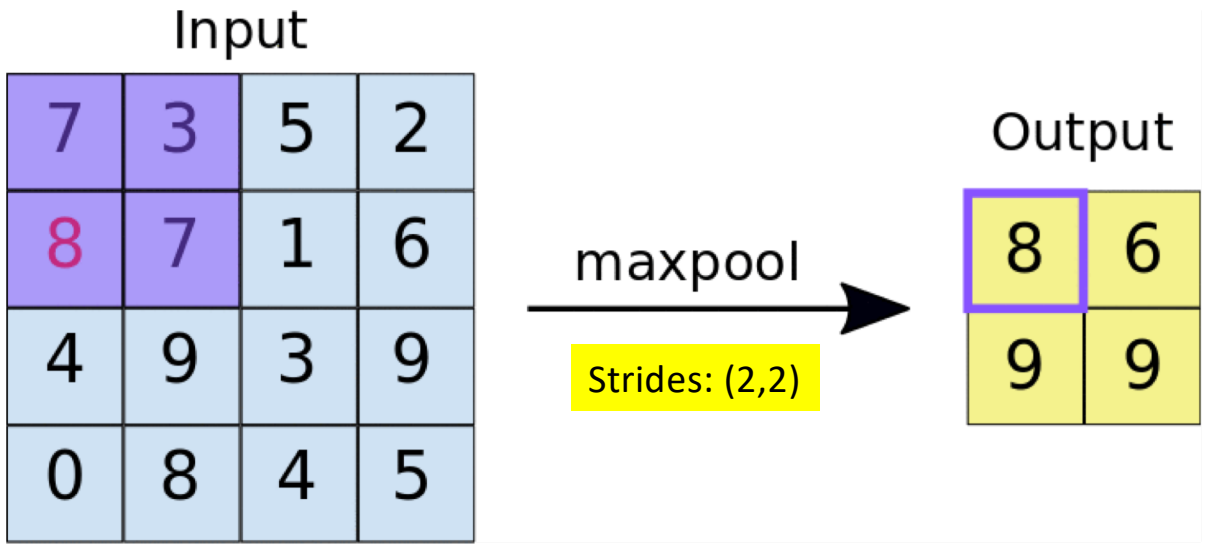
$Convolution\ parameters$:

$kernel_{height} \times kernel_{width} \times channel_{in} \times channel_{out}$

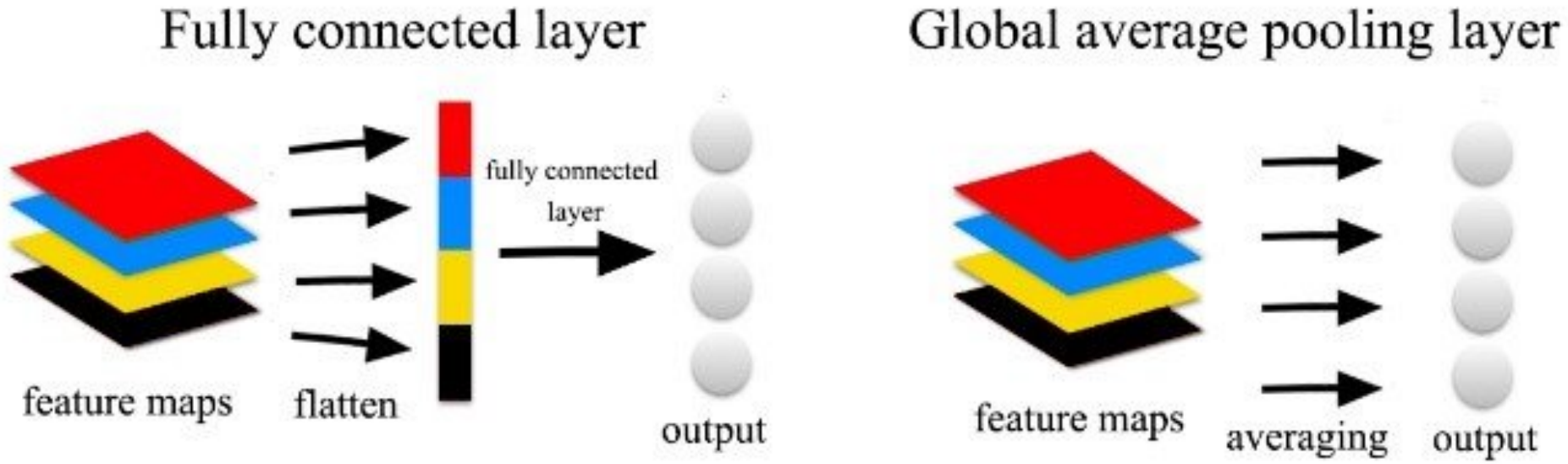
https://blog.csdn.net/Linli522362242/article/details/108414534

MaxPooling

- MaxPooling takes only maximum value of each region
- Helps in generalization as different inputs can map to same output



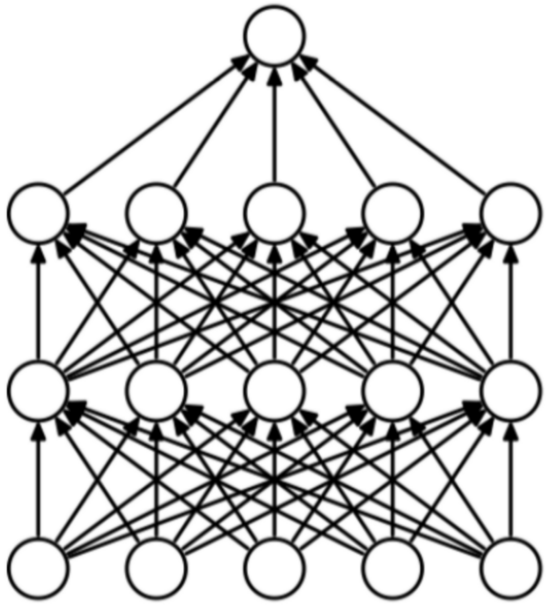
Going from features to output



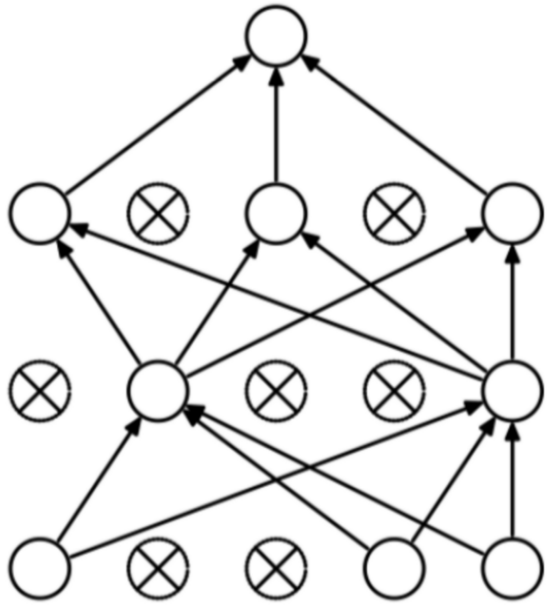
Real-Time Facial Affective Computing on Mobile Devices, Guo Yuanyuan (2020).

Dropout

- Training:
  - Keeps a proportion *p* of nodes/neurons randomly during training
    - Expected output value for activation value *x*:
      - *px + (1-p)(0) = px*
  - Scale the activation values by *1/p* during training to preserve same activation magnitude with original network
    - *px/p = x*
- Testing:
  - Do not drop any nodes/neurons
  - No scaling of activation values



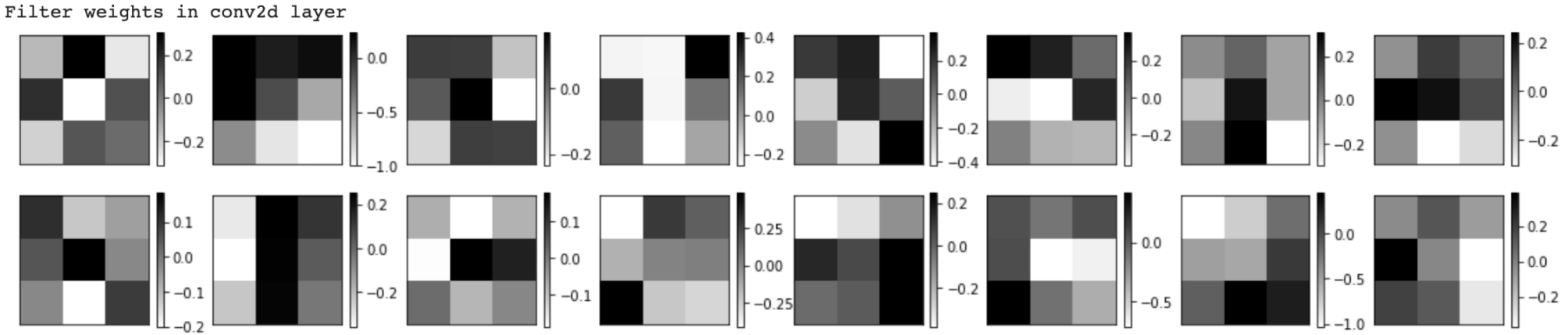
(a) Standard Neural Net



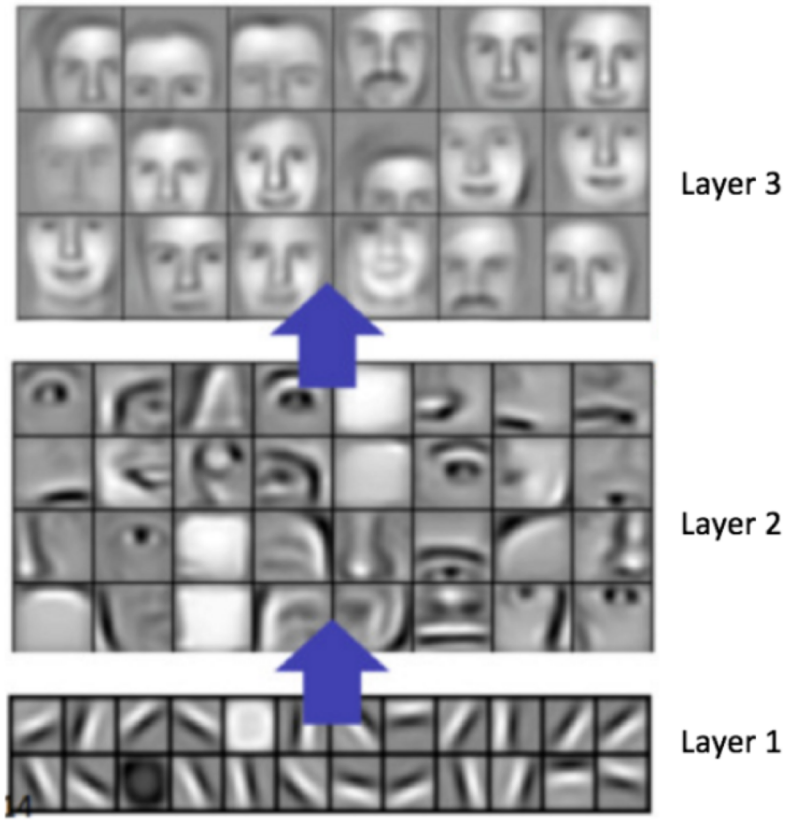
(b) After applying dropout.

https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

Visualizing Filter Weights



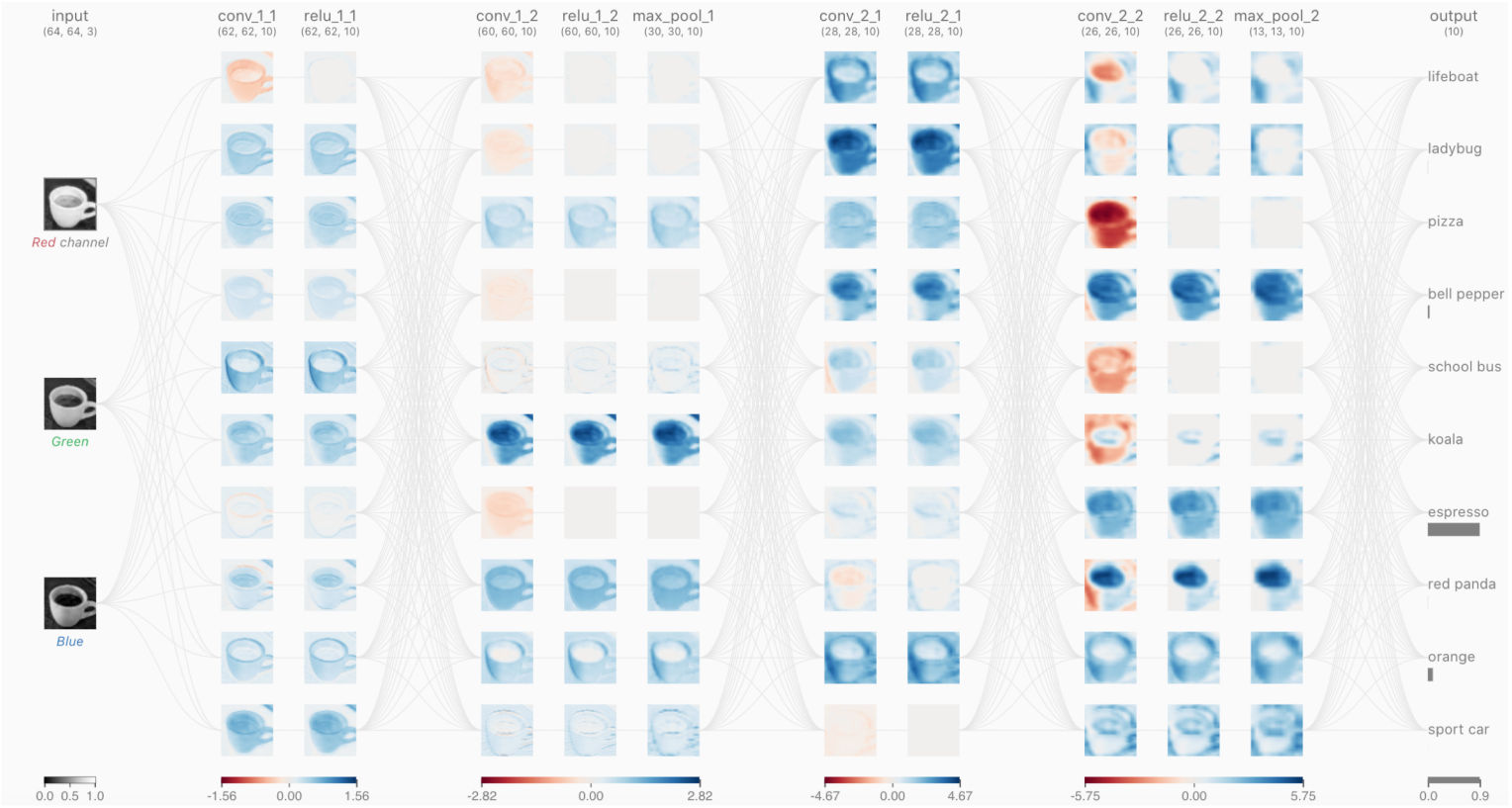
Pixel-to-features



https://web.eecs.umich.edu/~honglak/icml09-ConvolutionalDeepBeliefNetworks.pdf

CNN Explainer

- https://poloclub.github.io/cnn-explainer/

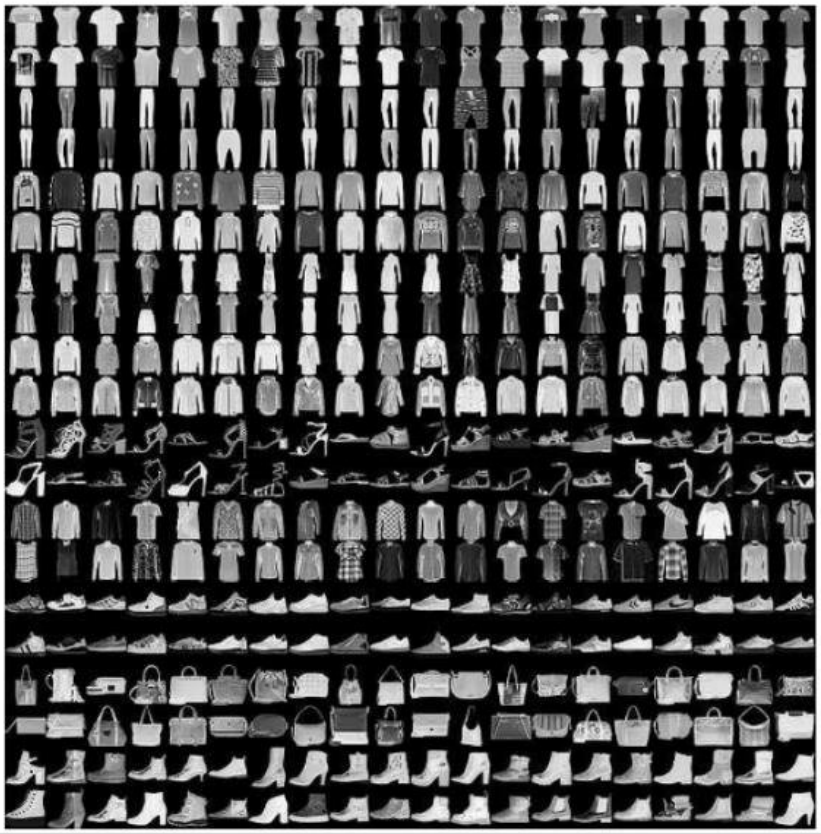


Fashion MNIST Classification

With CNN in TensorFlow

Fashion MNIST

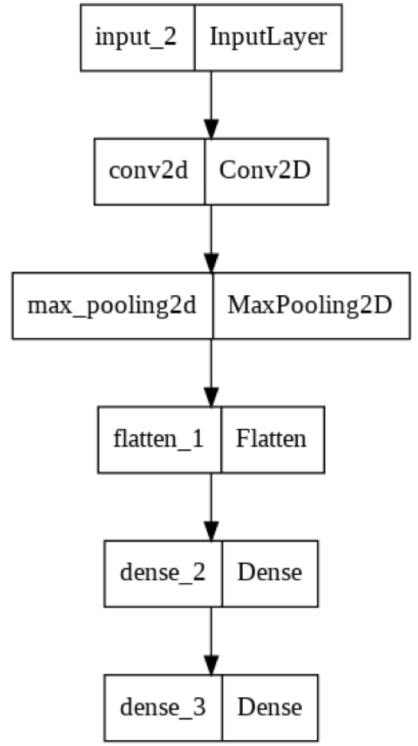
- 10 classes
- 60000 training images
- 10000 testing images
- Image Size: 28*28 pixels
  - (grayscale)

Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

TensorFlow – Define Model (Sequential)

- TensorFlow has many layers, like Lego blocks to build the model
- Sequential: Stack up Lego blocks in a line
  - Pipeline-style

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape = (28,28,1)),
    tf.keras.layers.Conv2D(filters=16, kernel_size=(3,3), strides=(1,1),
                           padding='same', activation = tf.nn.relu),
    tf.keras.layers.MaxPooling2D(pool_size = (2,2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation= tf.nn.relu),
    tf.keras.layers.Dense(10, activation = tf.nn.softmax)
])
```



TensorFlow – Define Model (Parameters)

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d (Conv2D)              (None, 28, 28, 16)        160

 max_pooling2d (MaxPooling2D  (None, 14, 14, 16)       0
 )

 flatten_1 (Flatten)          (None, 3136)              0

 dense_2 (Dense)              (None, 128)               401536

 dense_3 (Dense)              (None, 10)                1290

=================================================================
Total params: 402,986
Trainable params: 402,986
Non-trainable params: 0
```

- Conv2D takes in 1 channel, has 16 output channels, filter size 3x3
  - 1*16*3*3 = 144 weights
  - 16 filters = 16 biases
  - Total 160 parameters
- dense_2 has 3136*128 weights + 128 biases = 401536 parameters
- dense_3 has 128*10 weights + 10 biases = 1290 parameters
- Note: MaxPooling2D has no weights (why?)

TensorFlow - Overall

```
import tensorflow as tf
print(tf.__version__)

mnist = tf.keras.datasets.fashion_mnist

(training_images, training_labels) ,  (test_images, test_labels) = mnist.load_data()

training_images = training_images/255.0
test_images = test_images/255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape = (28,28,1)),
    tf.keras.layers.Conv2D(filters=16, kernel_size=(3,3), strides=(1,1), padding='same', activation = tf.nn.relu),
    tf.keras.layers.MaxPooling2D(pool_size = (2,2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation= tf.nn.relu),
    tf.keras.layers.Dense(10, activation = tf.nn.softmax)
])

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

model.fit(training_images.reshape(-1,28,28,1), training_labels, epochs=5)

model.evaluate(test_images, test_labels)
```

TensorFlow Practice

- Let us run the model!

Questions?