# CLGX-Hackathon 2019

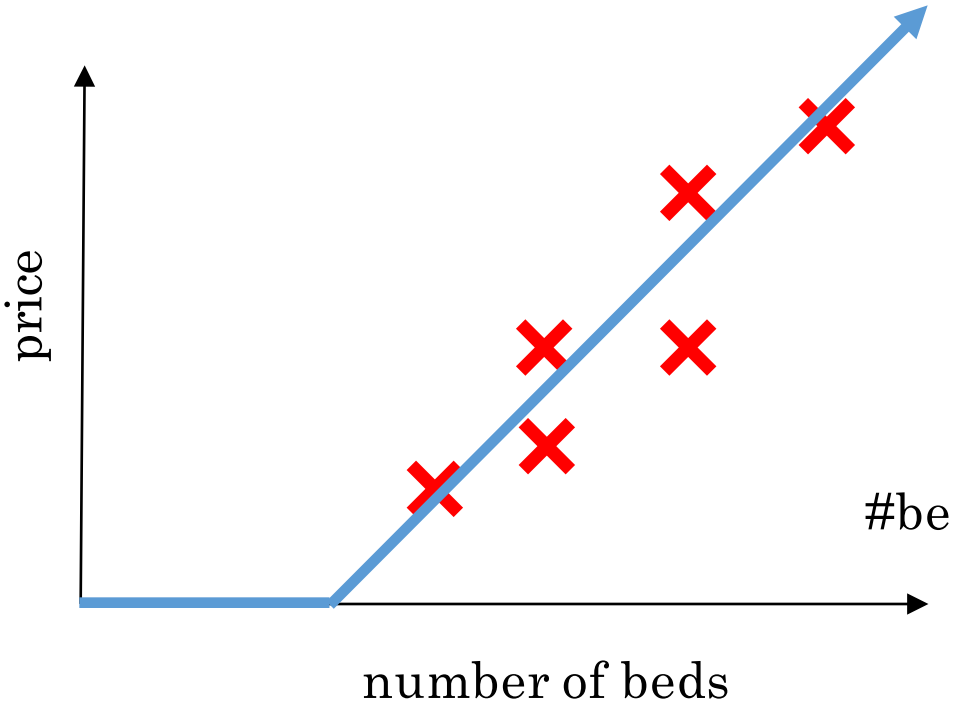
# Goals

- Not to introduce/explain all the concepts of deep learning. But to introduce a quick introduction, basic concepts of deep learning and tensorflow modules so that we can get out hands dirty on some image analysis as quick as possible.

# Agenda

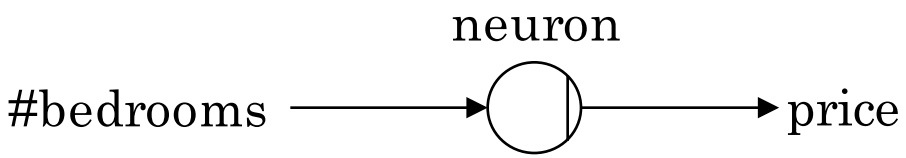
- Introduction to Deep Learning
  - Artificial Neural Networks (ANNs)
    - Activation functions
    - Optimizer
    - loss
  - Convolution 2D
  - MaxPooling 2D
  - Fully Connected
  - Dropout
  - Batch
- Demo programs with CIFAR dataset
  - ANNs
  - Convolution
  - Transfer Learning
- Assignment
  - Place 365 dataset
  - Requirements

# Housing Price Prediction



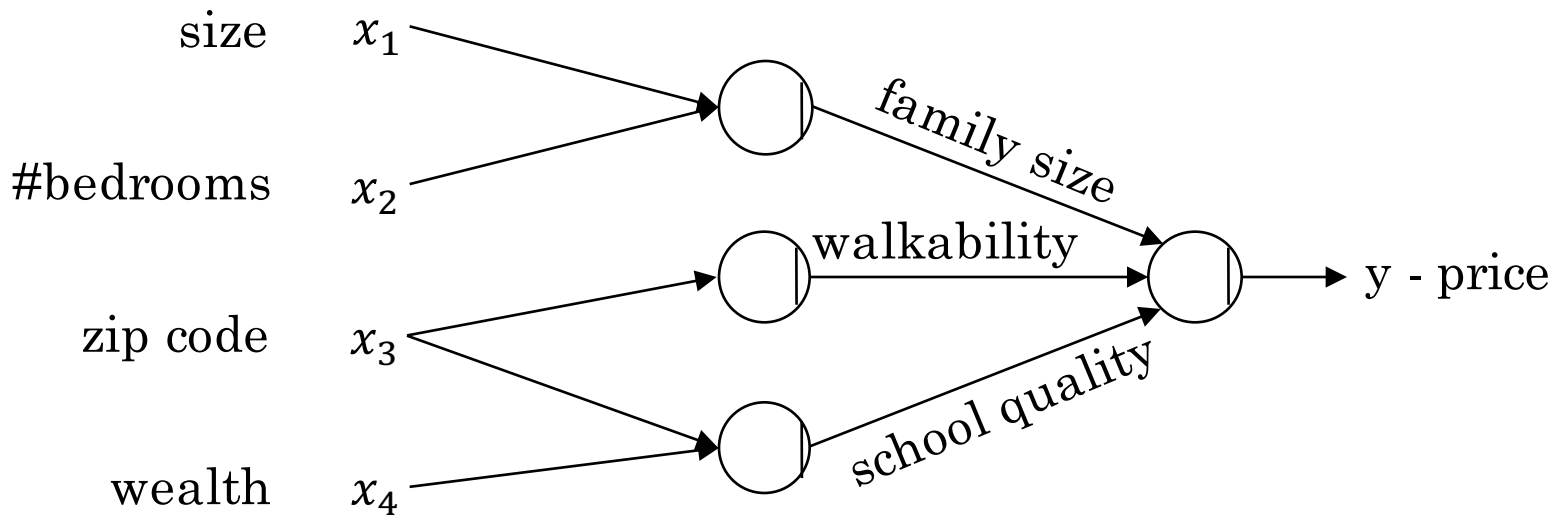

ReLU - Rectified Linear Unit

= max(0, x)

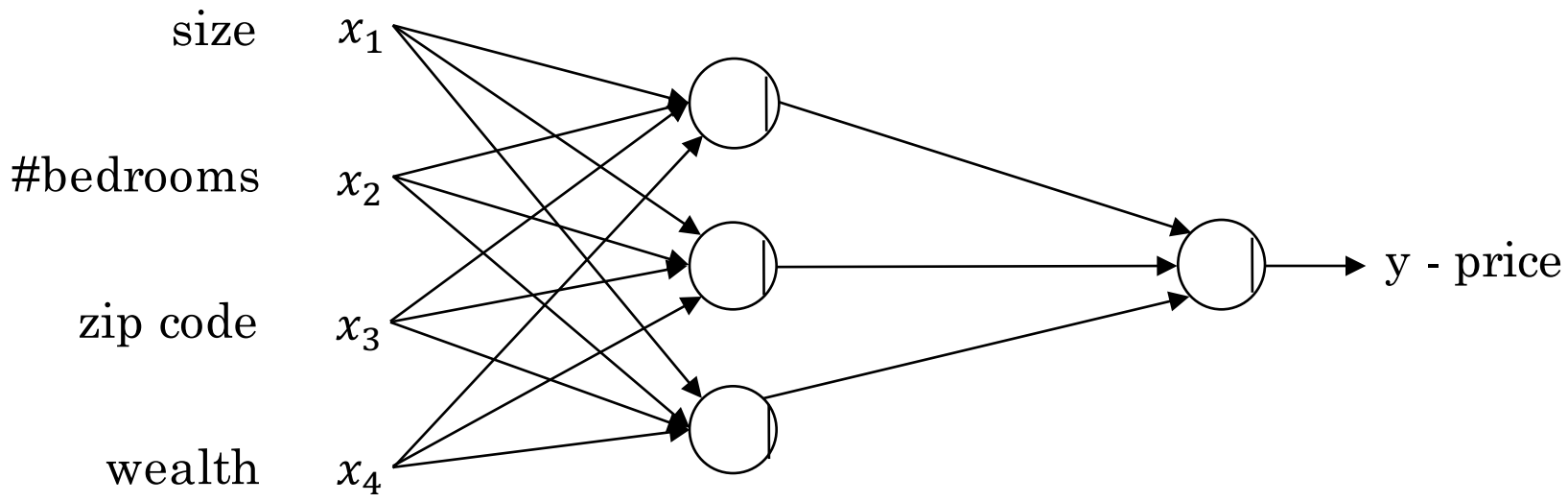



y – price
X - #bedrooms
w - weight
y = f(w * X + bias) = max(0, w * X + bias)

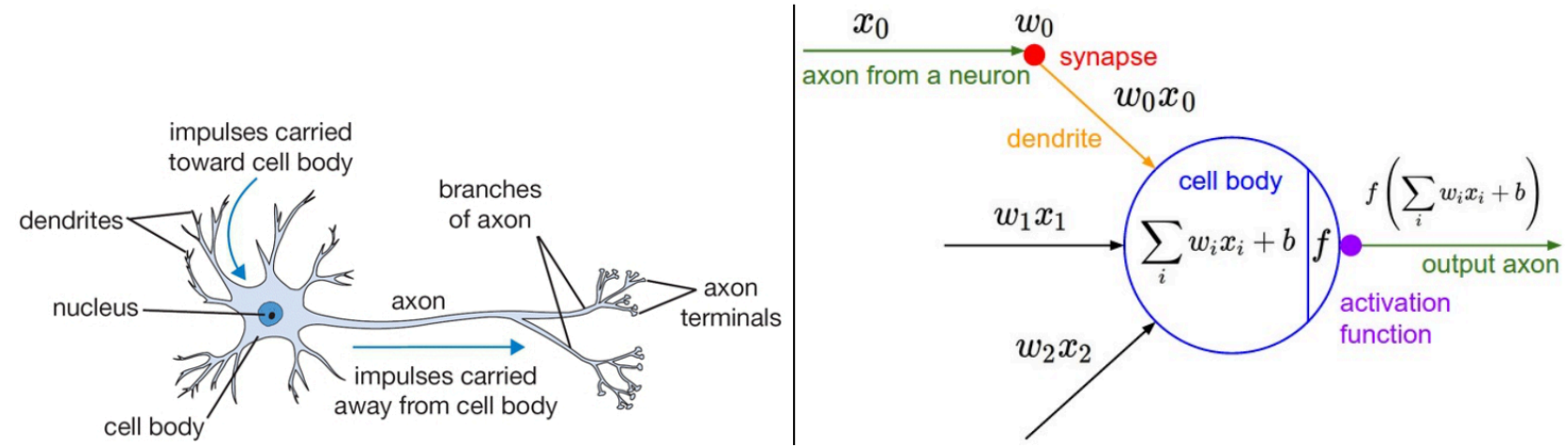
# Housing Price Prediction



# Housing Price Prediction

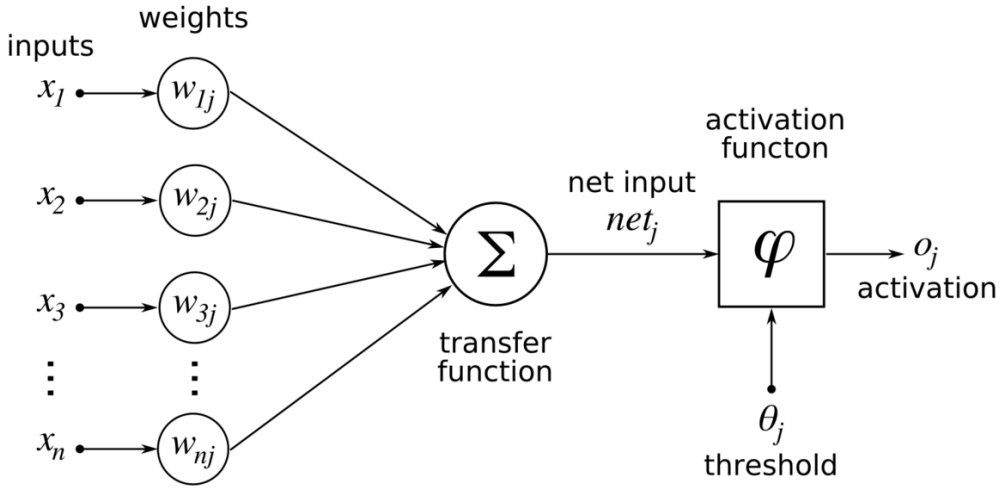


# How does activation function work?




A cartoon drawing of a biological neuron (left) and its mathematical model (right).

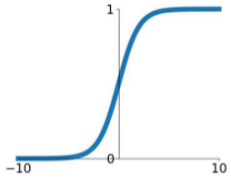
# How does activation function work?



# Activation Functions

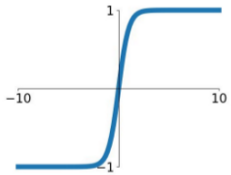
**Sigmoid**

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



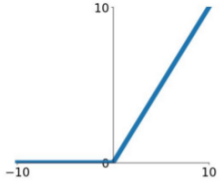

**tanh**

$$\tanh(x)$$



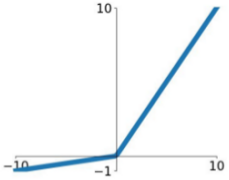

**ReLU**

$$\max(0, x)$$




**Leaky ReLU**

$$\max(0.1x, x)$$

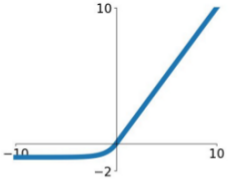



**Maxout**

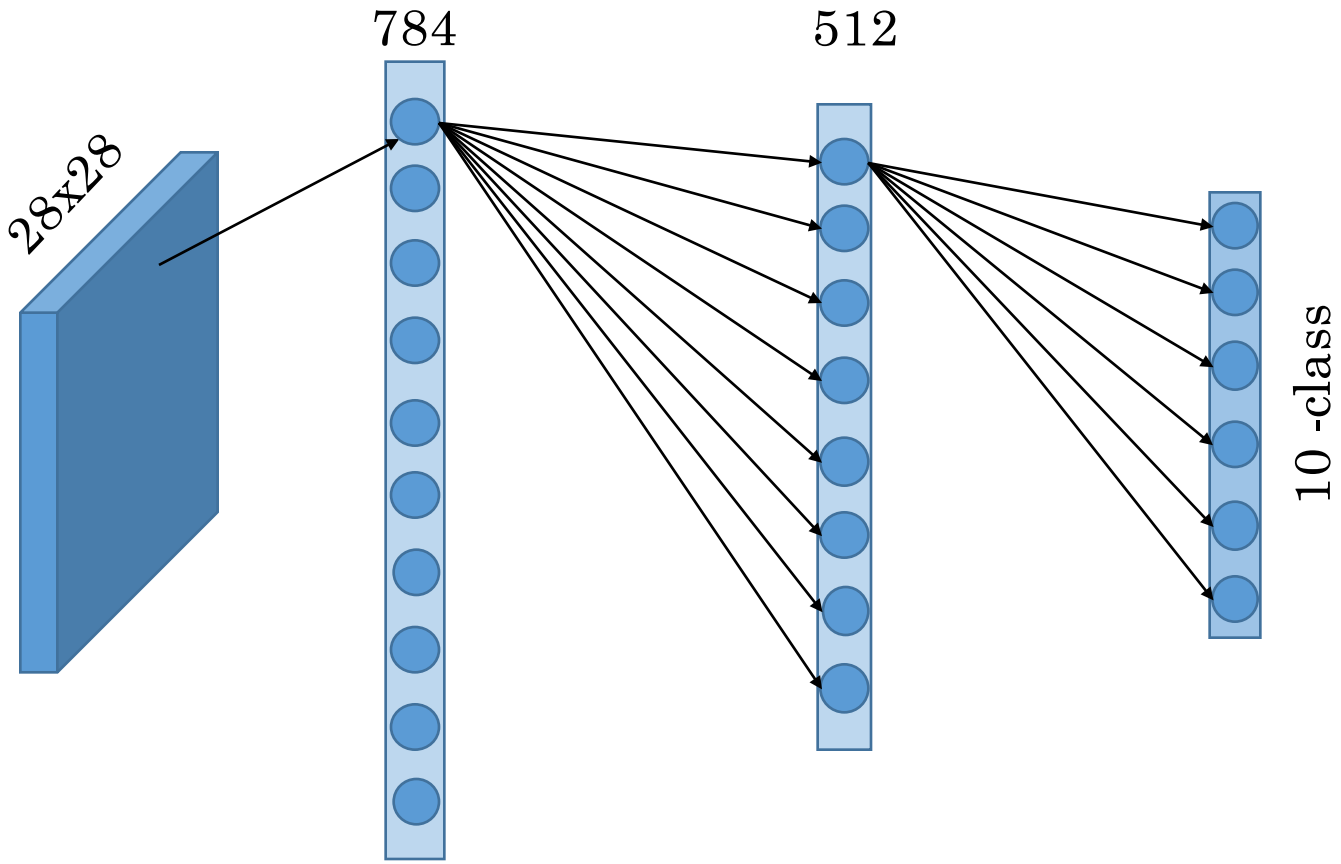
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

**ELU**

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



# ANN - mnist



# Loss and optimizer in tensorflow

| | loss | optimizer |
|---|---|---|
| Binary classification | • binary_crossentropy | • - sgd<br>• - adam<br>• - rmsprop<br>• - adagrad<br>• - adadelta<br>… |
| Mutilple classification | • sparse_categorical_crossentropy<br>• categorical_crossentropy | |
| Regression | • mean-absolute-error<br>• root-mean-square-error | |

# Feature Selection

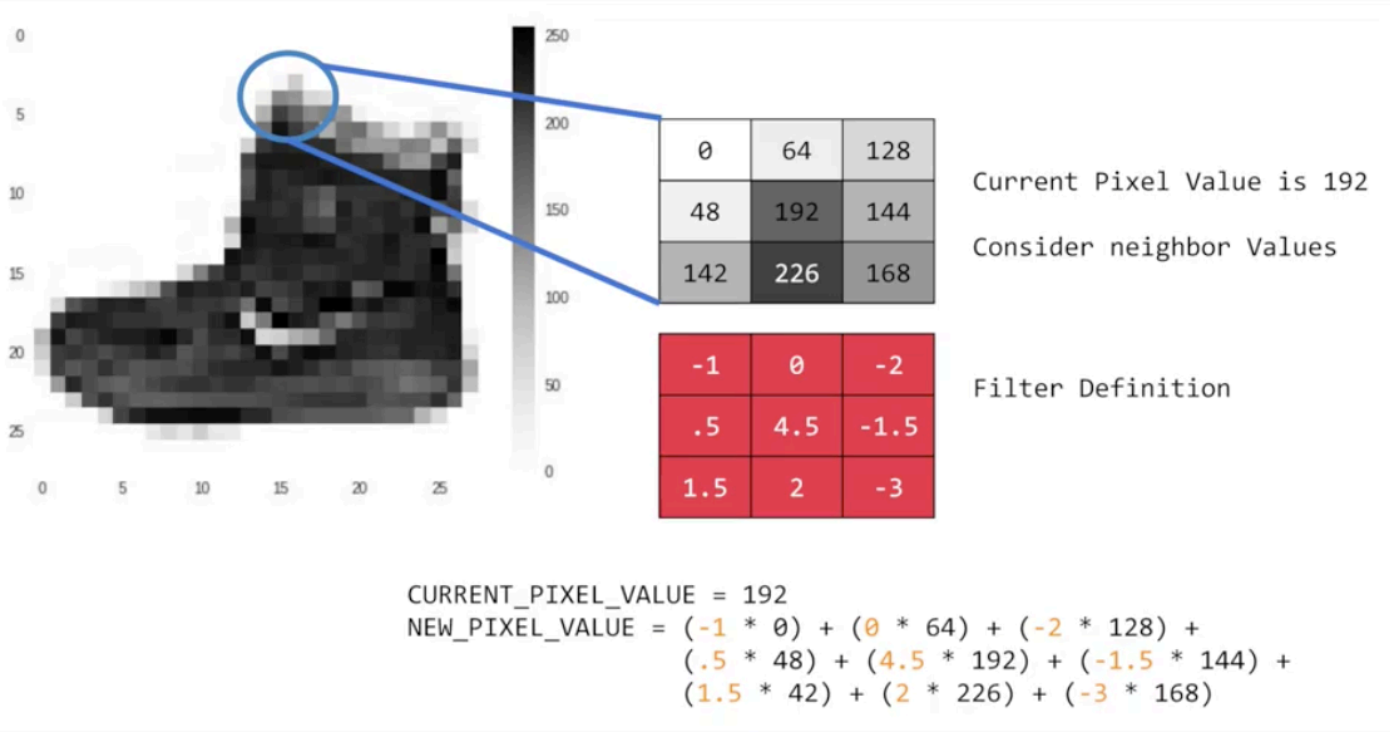
| | Technique |
|---|---|
| Linear regression | - Principal Component Analysis<br>- Bitwise selection<br>- -Chi-square test |
| Computer Vision | - ?<br>- ? |

# Mnist - dataset

- training: 60000 gray images
- validation: 10000 gray images
- each image size 28x28 with handwriting of a number
- label is the number in the image

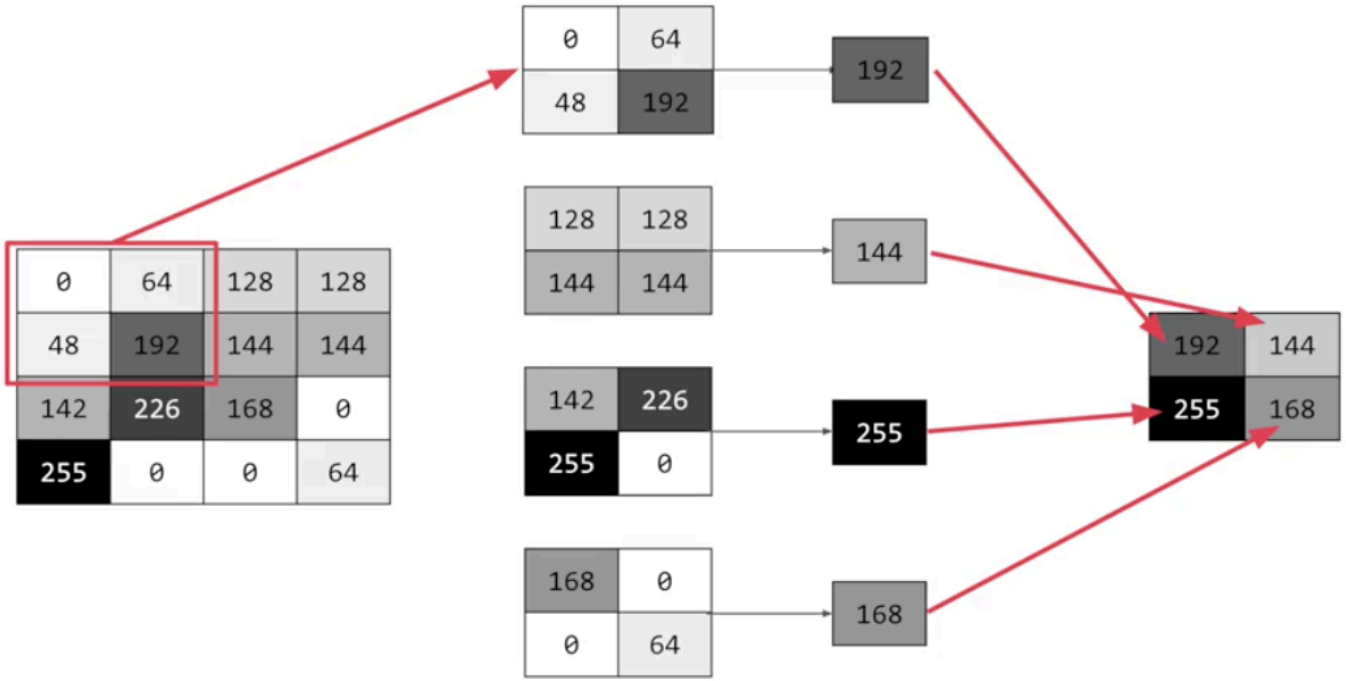
- Goal: build a ANN from training set to predict the number in image of validation set.

# Conv2D

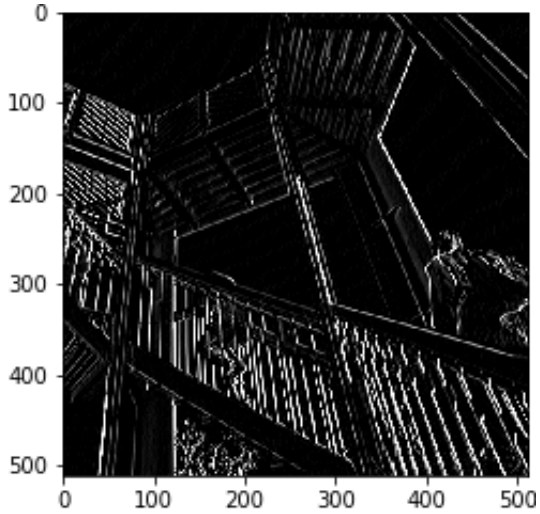


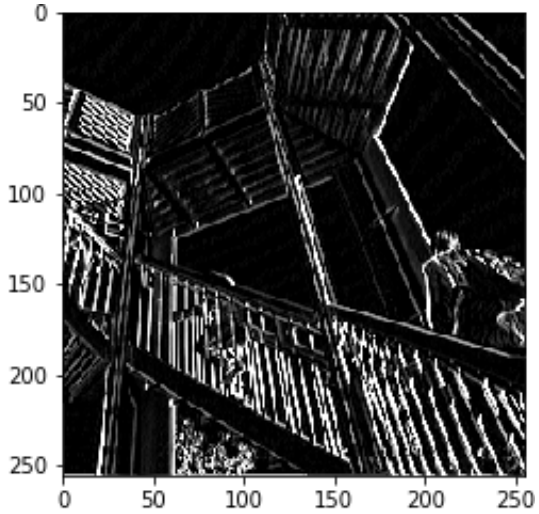

```
CURRENT_PIXEL_VALUE = 192
NEW_PIXEL_VALUE = (-1 * 0) + (0 * 64) + (-2 * 128) +
                  (.5 * 48) + (4.5 * 192) + (-1.5 * 144) +
                  (1.5 * 42) + (2 * 226) + (-3 * 168)
```

# MaxPool2D


0 64 128 128
48 192 144 144
142 226 168 0
255 0 0 64
0 64
48 192
192
128 128
144 144
144
142 226
255 0
255
168 0
0 64
168
192 144
255 168

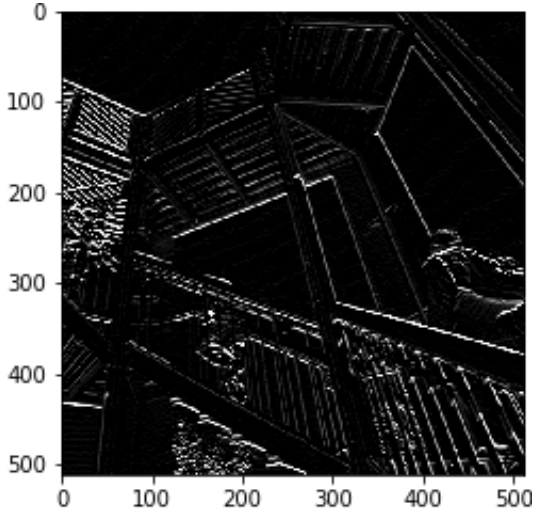
# Example of Conv2D and MaxPooling2D

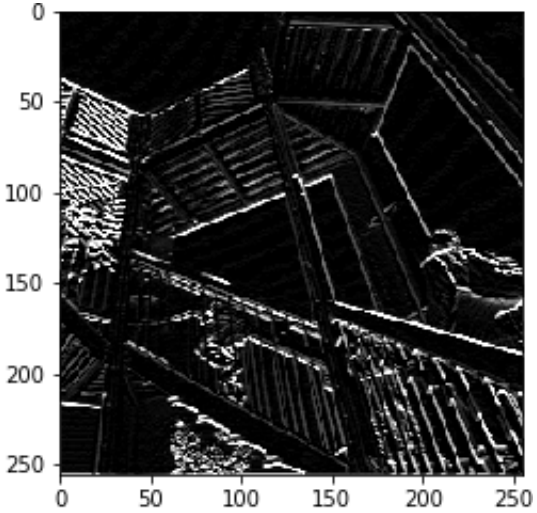



```
# vertical lines
# filter = [ [-1, -2, -1], [0, 0, 0], [1, 2, 1]]
```

# Example of Conv2D and MaxPooling2D

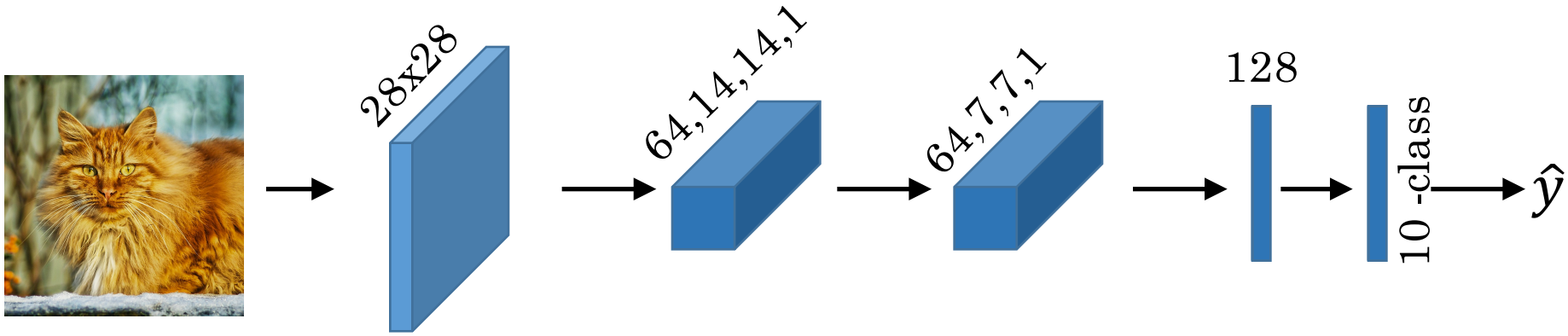



```
# horizontal lines
# filter = [ [-1, 0, 1], [-2, 0, 2], [-1, 0, 1]]
```

# Putting it together

- Training set $(x^{(1)}, y^{(1)}) \ldots (x^{(m)}, y^{(m)})$.



```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(64, (3,3), padding='same', activation='relu', input_shape=(28, 28, 1)),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Conv2D(64, (3,3), padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(2,2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(10, activation='softmax')
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

What is the default filter used by keras/tensorflow?

# BatchNormalization

# Batch

# CIFAR dataset

- test_vgg_cifar10_v2.ipynb

# Transfer Learning

# ImageGenerator

# How to prepare images data for deep learning