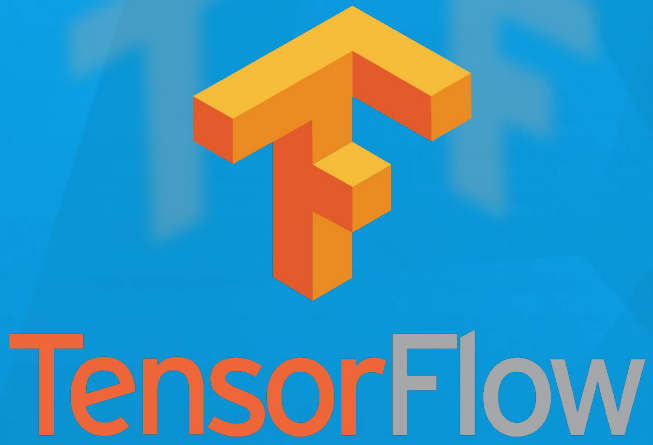
# TensorFlow 2.x and Keras

Learn how to leverage features that help you achieve results in AI




# What's in store for today?

1. **Introduction to Machine Learning/Deep Learning landscape**
   - a. The classification problem
   - b. Simple NLP application
2. **Back to the basics**
   - a. Linear Regression in TF
   - b. Leveraging high level Keras API

# Image Classification

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 08 | 02 | 22 | 97 | 38 | 15 | 00 | 40 | 00 | 75 | 04 | 05 | 07 | 78 | 52 | 12 | 50 | 77 | 91 | 08 |
| 49 | 49 | 99 | 40 | 17 | 81 | 18 | 57 | 60 | 87 | 17 | 40 | 98 | 43 | 69 | 48 | 04 | 56 | 62 | 00 |
| 81 | 49 | 31 | 73 | 55 | 79 | 14 | 29 | 93 | 71 | 40 | 67 | 53 | 88 | 30 | 03 | 49 | 13 | 36 | 65 |
| 52 | 70 | 95 | 23 | 04 | 60 | 11 | 42 | 69 | 24 | 68 | 56 | 01 | 32 | 56 | 71 | 37 | 02 | 36 | 91 |
| 22 | 31 | 16 | 71 | 51 | 67 | 63 | 89 | 41 | 92 | 36 | 54 | 22 | 40 | 40 | 28 | 66 | 33 | 13 | 80 |
| 24 | 47 | 32 | 60 | 99 | 03 | 45 | 02 | 44 | 75 | 33 | 53 | 78 | 36 | 84 | 20 | 35 | 17 | 12 | 50 |
| 32 | 98 | 81 | 28 | 64 | 23 | 67 | 10 | 26 | 38 | 40 | 67 | 59 | 54 | 70 | 66 | 18 | 38 | 64 | 70 |
| 67 | 26 | 20 | 68 | 02 | 62 | 12 | 20 | 95 | 63 | 94 | 39 | 63 | 08 | 40 | 91 | 66 | 49 | 94 | 21 |
| 24 | 55 | 58 | 05 | 66 | 73 | 99 | 26 | 97 | 17 | 78 | 78 | 96 | 83 | 14 | 88 | 34 | 89 | 63 | 72 |
| 21 | 36 | 23 | 09 | 75 | 00 | 76 | 44 | 20 | 45 | 35 | 14 | 00 | 61 | 33 | 97 | 34 | 31 | 33 | 95 |
| 78 | 17 | 53 | 28 | 22 | 75 | 31 | 67 | 15 | 94 | 03 | 80 | 04 | 62 | 16 | 14 | 09 | 53 | 56 | 92 |
| 16 | 39 | 05 | 42 | 96 | 35 | 31 | 47 | 55 | 58 | 88 | 24 | 00 | 17 | 54 | 24 | 36 | 29 | 85 | 57 |
| 86 | 56 | 00 | 48 | 35 | 71 | 89 | 07 | 05 | 44 | 44 | 37 | 44 | 60 | 21 | 58 | 51 | 54 | 17 | 58 |
| 19 | 80 | 81 | 68 | 05 | 94 | 47 | 69 | 28 | 73 | 92 | 13 | 86 | 52 | 17 | 77 | 04 | 89 | 55 | 40 |
| 04 | 52 | 08 | 83 | 97 | 35 | 99 | 16 | 07 | 97 | 57 | 32 | 16 | 26 | 26 | 79 | 33 | 27 | 98 | 66 |
| 88 | 36 | 68 | 87 | 57 | 62 | 20 | 72 | 03 | 46 | 33 | 67 | 46 | 55 | 12 | 32 | 63 | 93 | 53 | 69 |
| 04 | 42 | 16 | 73 | 38 | 25 | 39 | 11 | 24 | 94 | 72 | 18 | 08 | 46 | 29 | 32 | 40 | 62 | 76 | 36 |
| 20 | 69 | 36 | 41 | 72 | 30 | 23 | 88 | 34 | 62 | 99 | 69 | 82 | 67 | 59 | 85 | 74 | 04 | 36 | 16 |
| 20 | 73 | 35 | 29 | 78 | 31 | 90 | 01 | 74 | 31 | 49 | 71 | 48 | 86 | 81 | 16 | 23 | 57 | 05 | 54 |
| 01 | 70 | 54 | 71 | 83 | 51 | 54 | 69 | 16 | 92 | 33 | 48 | 61 | 43 | 52 | 01 | 89 | 19 | 67 | 48 |

What the computer sees

Image classification

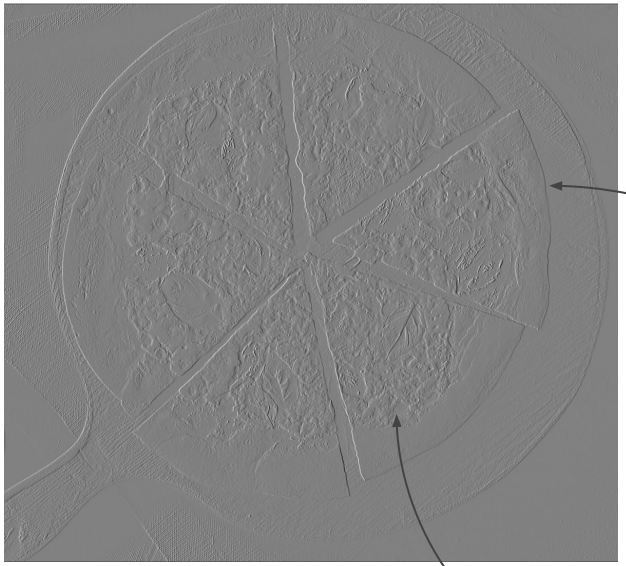
70% Pizza

20% Momo

7% Pasta

3% Spaghetti

# Algorithmic Approach




Round pizza square box cut approx. in triangles

# Data-Driven Approach

1. Collect a dataset of image and labels
2. Use Machine Learning to train an image classifier
3. Evaluate the classifier on a withheld set of test images

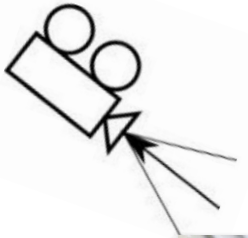


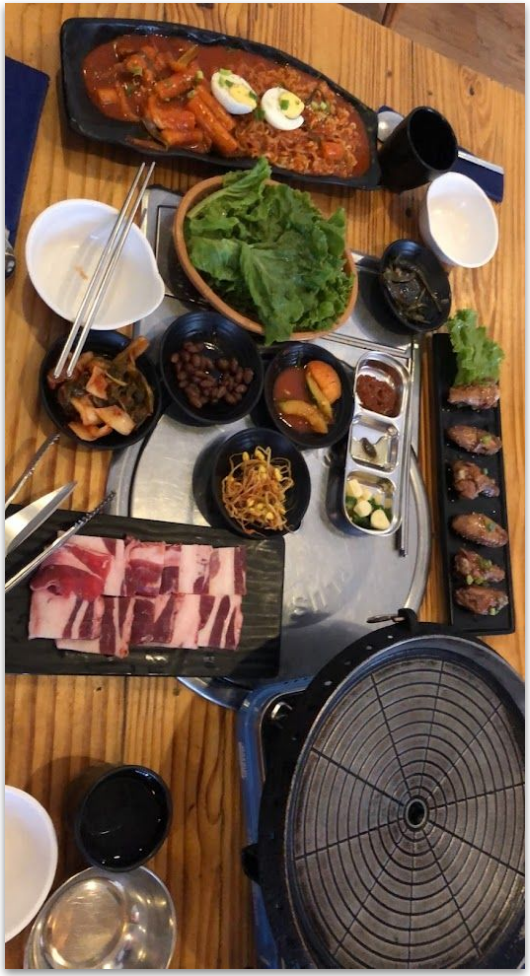







# Challenges: Viewpoint Variation

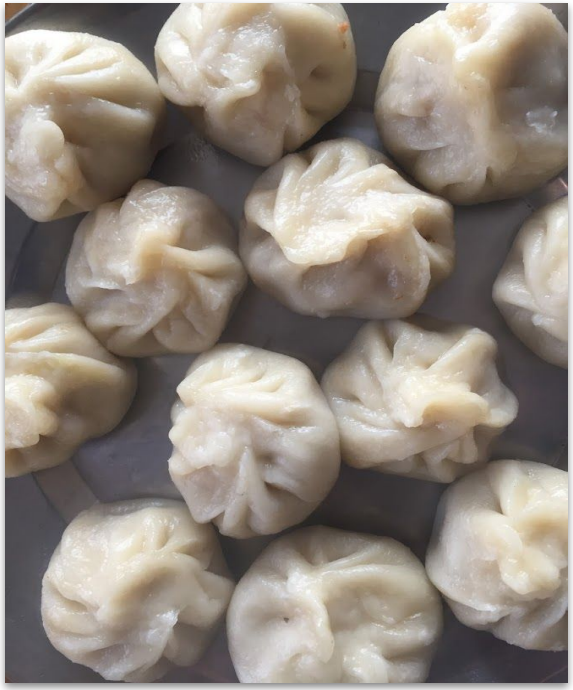




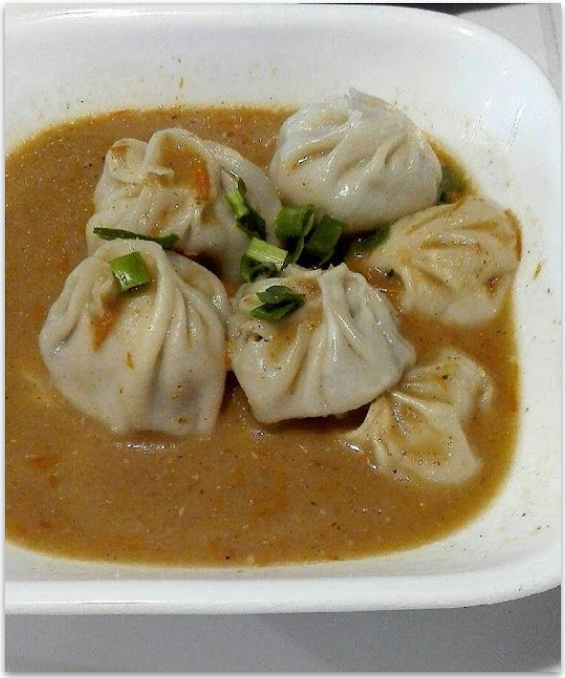
# Challenges: Illumination

# Challenges: Deformation

# Challenges: Occlusion



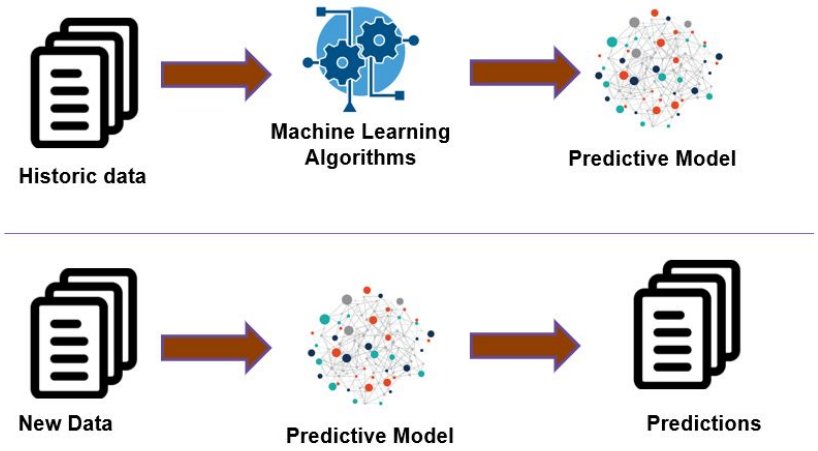
# Challenges: Intra-class Variation



# Pseudocode : Image Classification

```
def train(train_images, train_labels):
    # build a model for images -> labels
    return model

def predict(model, image):
    # predict label using the model
    return class_label
```



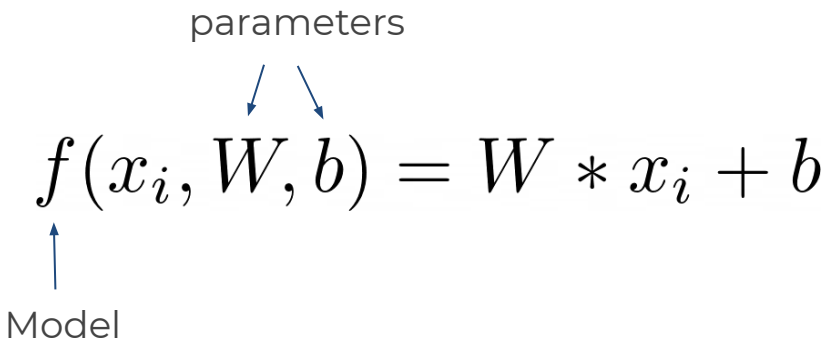

[How hard is it to use a State-of-the-Art pretrained image classifier?]

# Pseudocode : NLP use case

How hard is it to use a State-of-the-Art pretrained question answering system?


# Programming in **TensorFlow Core**
# Vs
# Using high-level API : **Keras**

# Linear Regressor



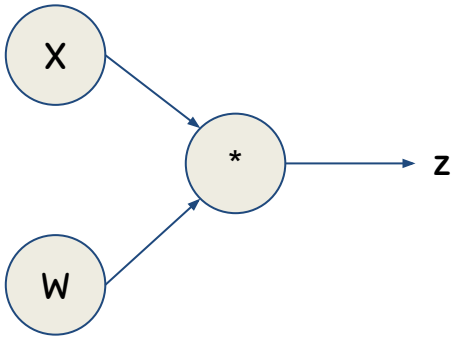

- **Loss function:** quantifying what it means to have a good "W"
- **Optimization:** start with random W and find a W that minimizes the loss

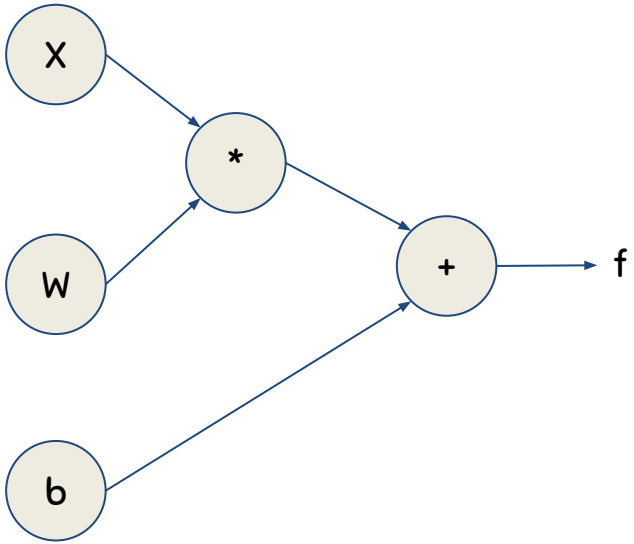
# Toy Example

$$F = C * 1.8 + 32$$

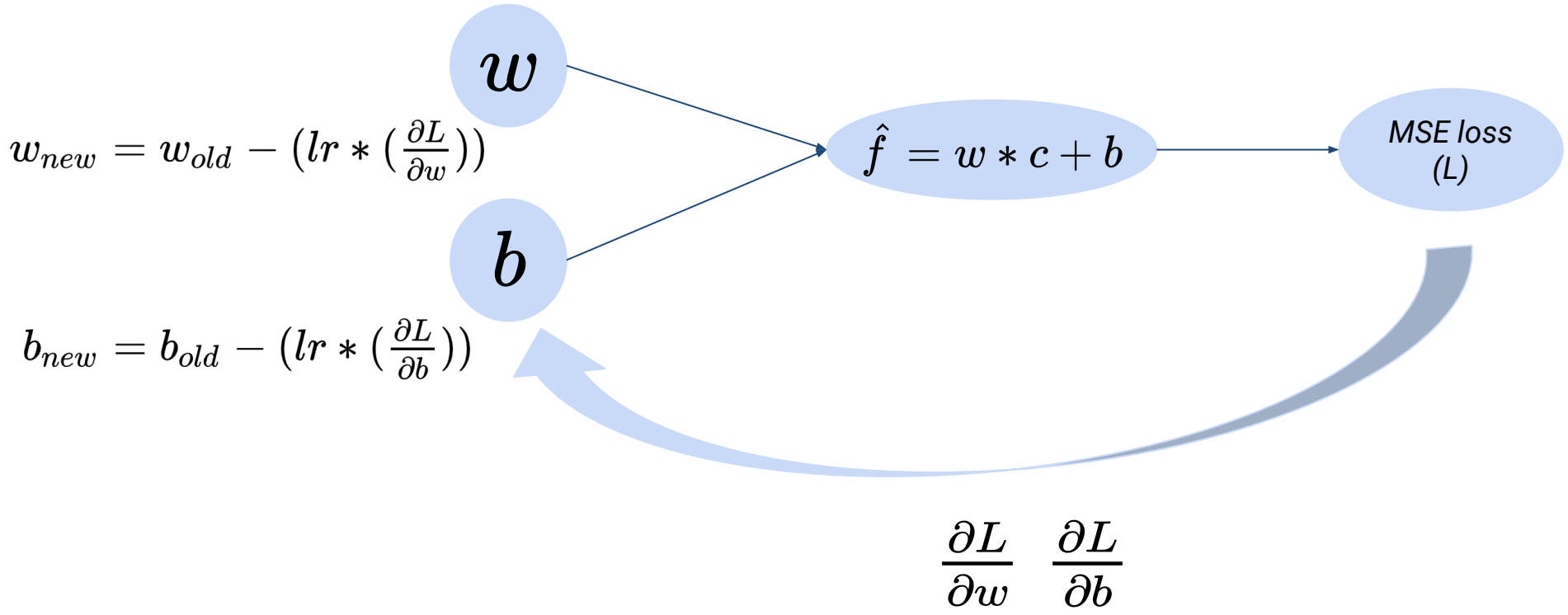
| | | | | | |
|---|---|---|---|---|---|
| **Celsius(°C)** | -40 | 0 | 22 | 37 | 100 |
| **Fahrenheit(°F)** | -40 | 32 | 77 | 98.6 | 212 |

# Computational Graph



# Linear Regression flow



# Keras Sequential API

```
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential

model = Sequential()

model.add(Dense(units=64, activation='relu'))
model.add(Dense(units=10, activation='softmax'))

model.compile(...)
model.fit(...)
model.evaluate(...)
model.predict(...)
```

# Keras Functional API

```
from tensorflow.keras.layers import Dense

inputs = keras.Input(shape=(784,))

dense = Dense(64, activation="relu")
x = dense(inputs)

x = Dense(64, activation="relu")(x)
outputs = layers.Dense(10)(x)

model = keras.Model(inputs=inputs, outputs=outputs, name="my_model")

model.compile(...)
model.fit(...)
model.evaluate(...)
model.predict(...)
```

# Keras Subclassing Method

```
class Linear(keras.layers.Layer):
    def __init__(self, units=32, input_dim=32):
        super(Linear, self).__init__()
        self.w = self.add_weight(
            shape=(input_dim, units), initializer="random_normal", trainable=True
        )
        self.b = self.add_weight(shape=(units,), initializer="zeros", trainable=True)

    def call(self, inputs):
        return tf.matmul(inputs, self.w) + self.b


x = tf.ones((2, 2))
linear_layer = Linear(4, 2)
y = linear_layer(x)
print(y)
```

# TensorFlow 2.x : Ecosystem