AI Workshop - Images

Asst. Prof. Dr. Niwan Wattanakitrungroj

School of Information Technology,
King Mongkut's University of Technology Thonburi

Outline

1 Intro. of AI

2 Images

3 CNN

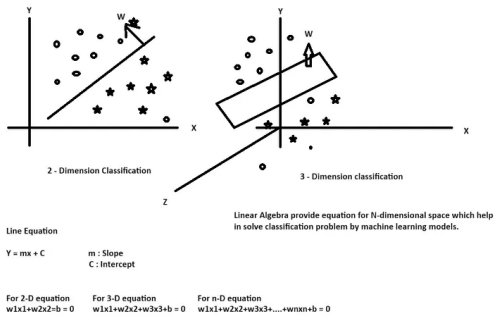
4 Objection Detection

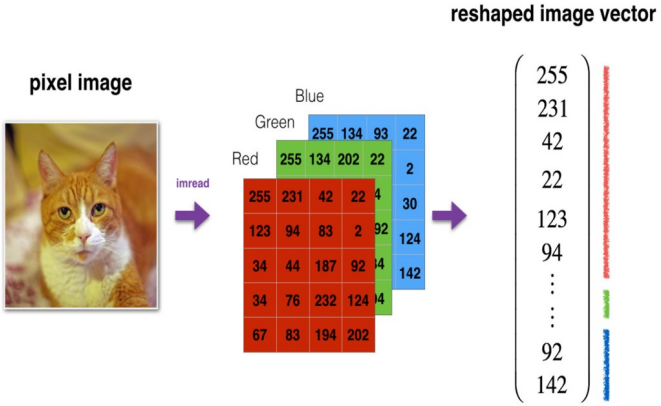
What is Artificial Intelligence?

Artificial Intelligence (AI) refers to the ability of a computer or machine to perform tasks that would normally require human-like intelligence, such as learning, problem solving, decision making, and language understanding.

Essential Mathematics for AI

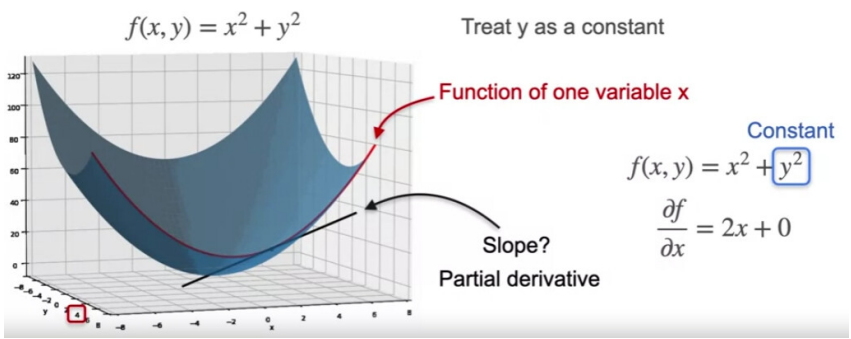
- Linear Algebra: for understanding data structures and transformations
- Calculus: for modeling changes and optimizing functions
- Optimization: for enhancing model performance
- Probability: for making informed decisions under uncertainty

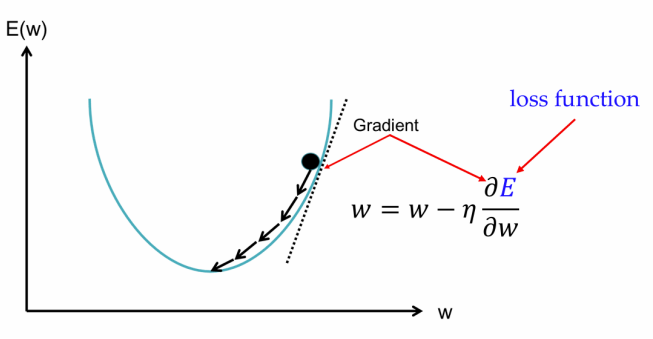




Essential Mathematics for AI

- Linear Algebra: for understanding data structures and transformations
- Calculus: for modeling changes and optimizing functions
- Optimization: for enhancing model performance
- Probability: for making informed decisions under uncertainty





$$w_i \leftarrow w_i + \Delta w_i$$

Where,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

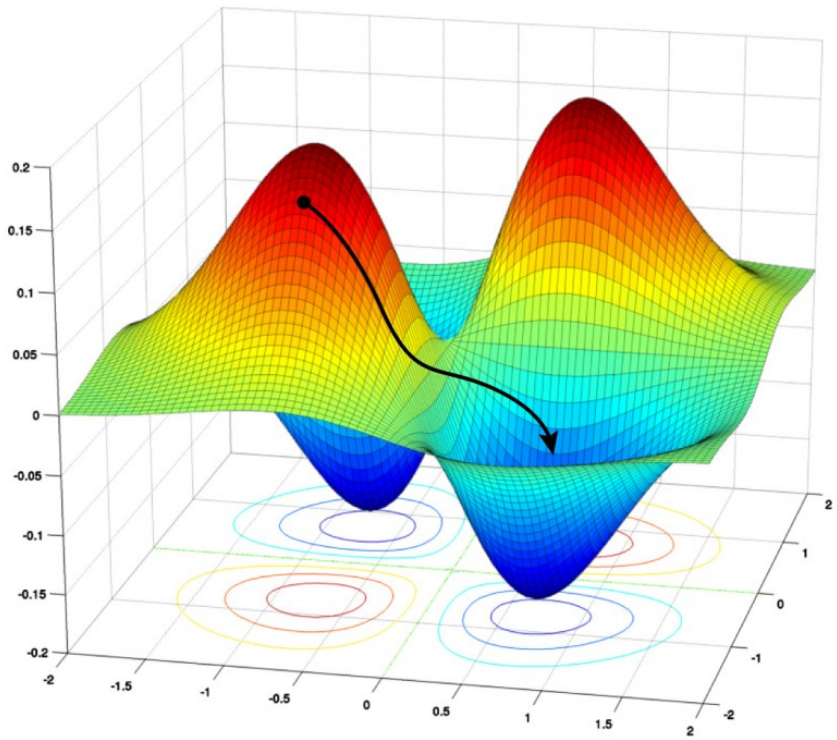
and

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

t_d = actual target value at a data point, d *and* O_d = predicted output values of d

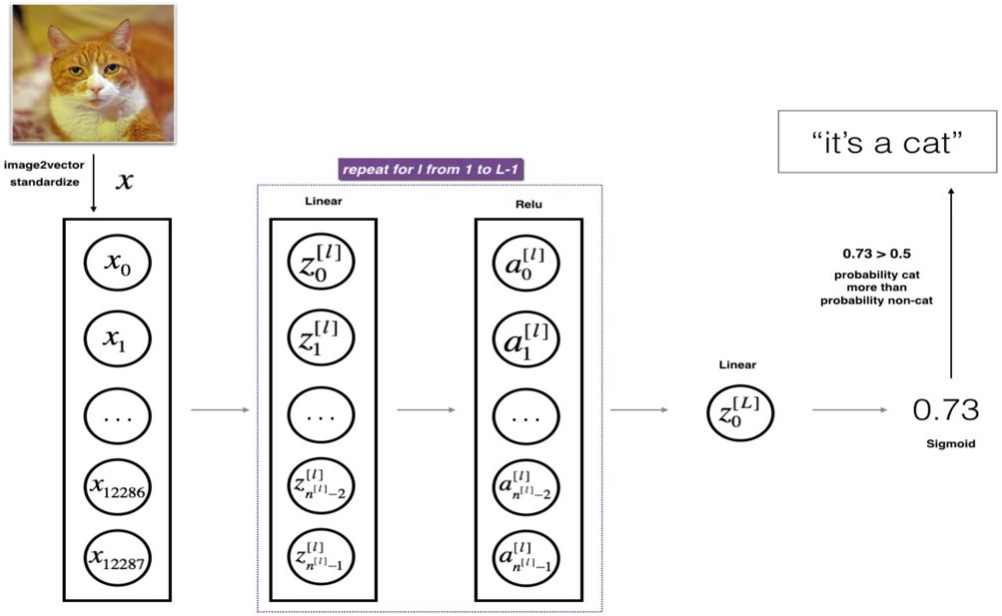
Essential Mathematics for AI

- Linear Algebra: for understanding data structures and transformations
- Calculus: for modeling changes and optimizing functions
- Optimization: for enhancing model performance
- Probability: for making informed decisions under uncertainty



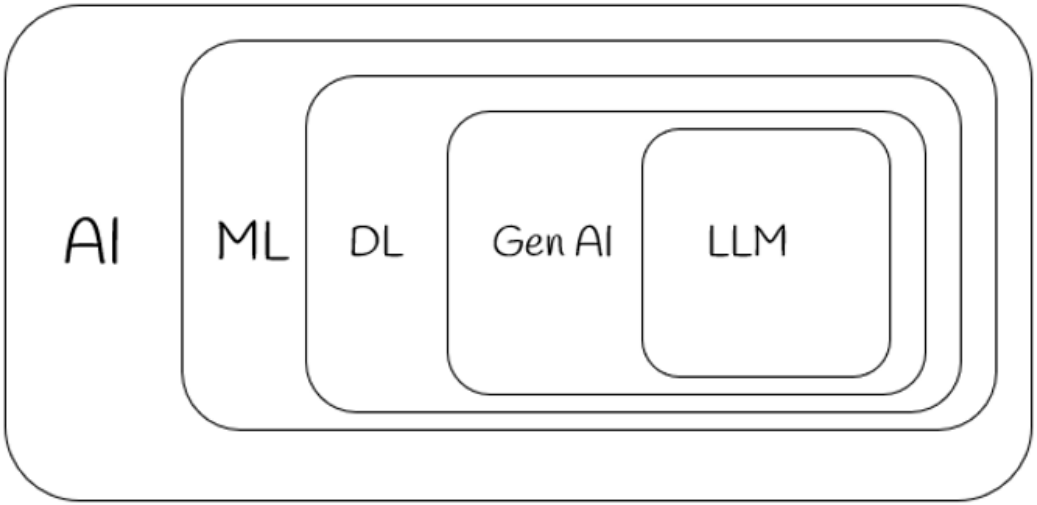
Essential Mathematics for AI

- Linear Algebra: for understanding data structures and transformations
- Calculus: for modeling changes and optimizing functions
- Optimization: for enhancing model performance
- Probability: for making informed decisions under uncertainty



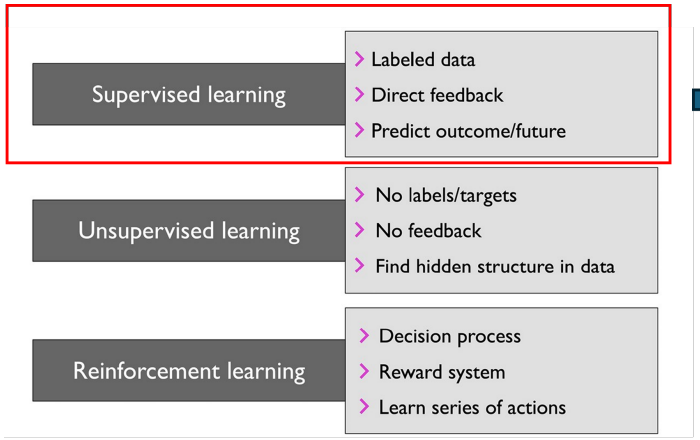
Subfield of AI

- Machine Learning (ML)
- Deep Learning (DL)
- Generative AI (Gen AI)
- Large Language Models (LLM)

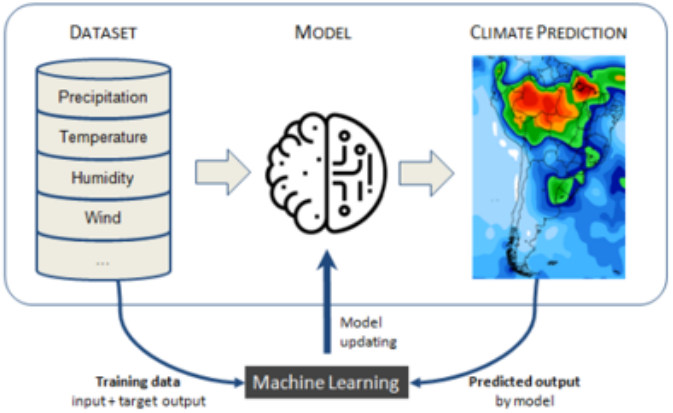


Subfield of AI : Machine Learning (ML)

Machine Learning (ML) is a subfield of AI that focuses on the development of algorithms and models that can learn from and make predictions on data.

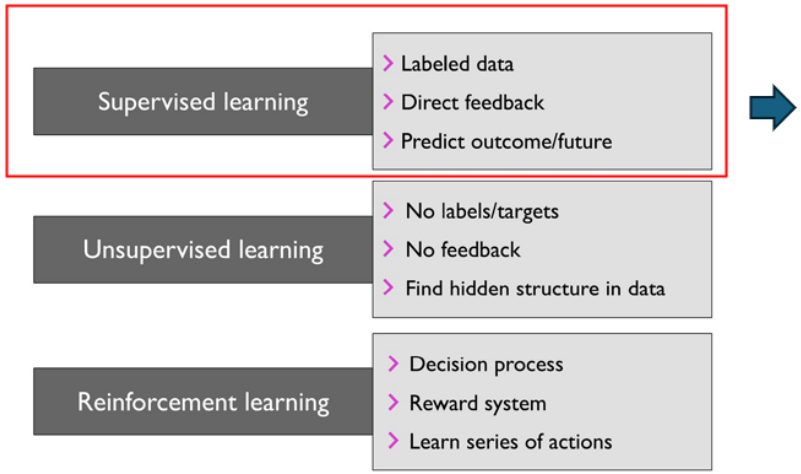


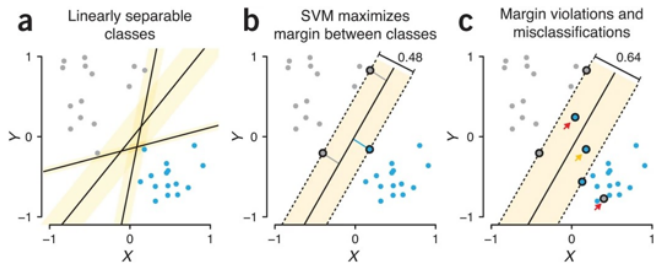
Features					Label
Position	Experience	Skill	Country	City	Salary ($)
Developer	0	1	USA	New York	103100
Developer	1	1	USA	New York	104900
Developer	2	1	USA	New York	106800
Developer	3	1	USA	New York	108700
Developer	4	1	USA	New York	110400
Developer	5	1	USA	New York	112300
Developer	6	1	USA	New York	114200
Developer	7	1	USA	New York	116100
Developer	8	1	USA	New York	117800
Developer	9	1	USA	New York	119700
Developer	10	1	USA	New York	121600

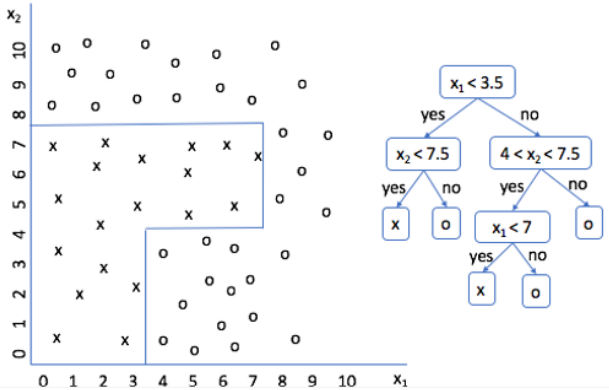


Subfield of AI : Machine Learning (ML)

Machine Learning (ML) is a subfield of AI that focuses on the development of algorithms and models that can learn from and make predictions on data.

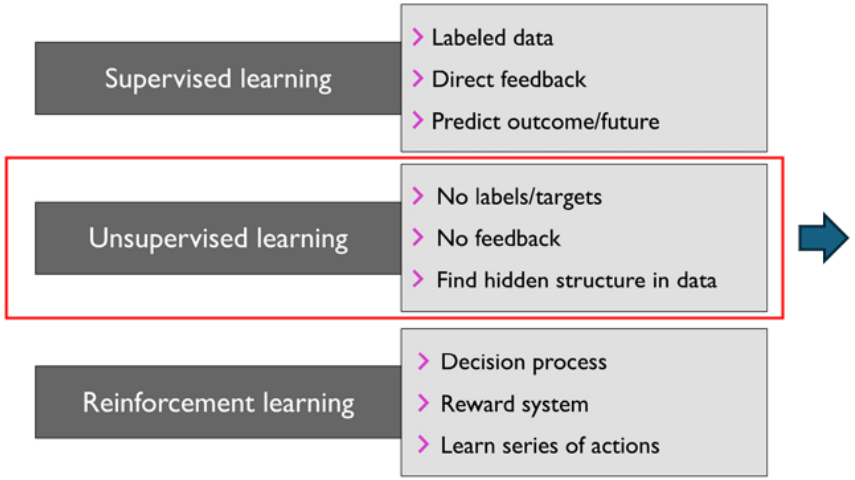


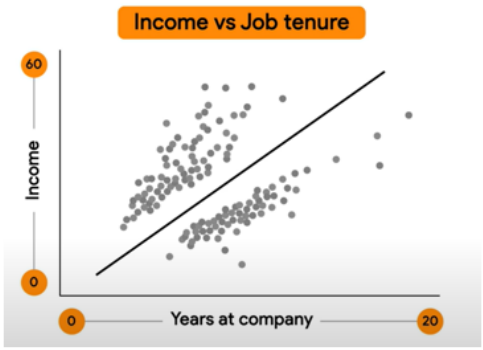




Subfield of AI : Machine Learning (ML)

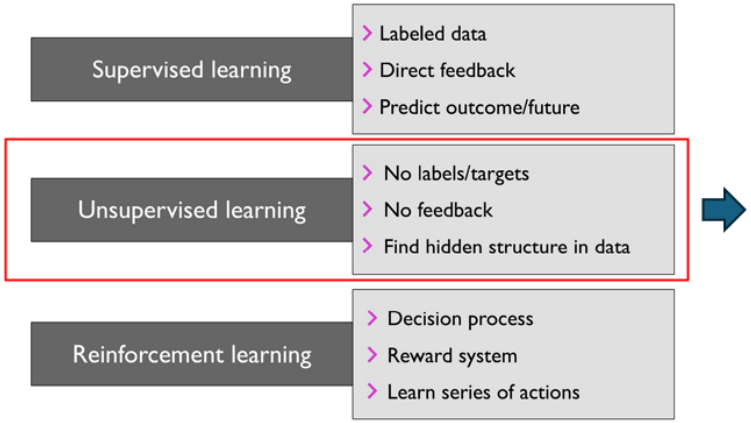
Machine Learning (ML) is a subfield of AI that focuses on the development of algorithms and models that can learn from and make predictions on data.

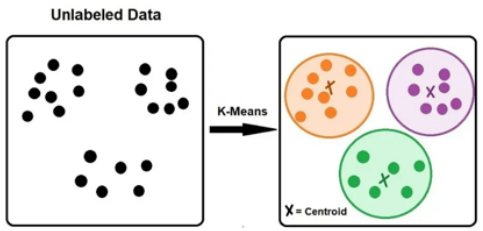


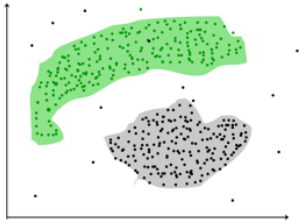


Subfield of AI : Machine Learning (ML)

Machine Learning (ML) is a subfield of AI that focuses on the development of algorithms and models that can learn from and make predictions on data.

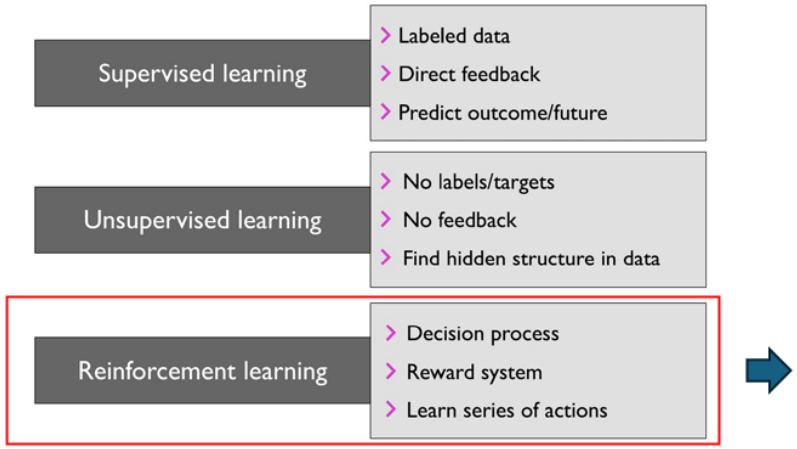


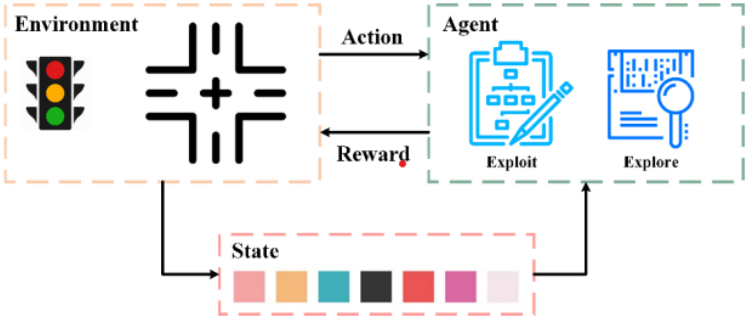




Subfield of AI : Machine Learning (ML)

Machine Learning (ML) is a subfield of AI that focuses on the development of algorithms and models that can learn from and make predictions on data.

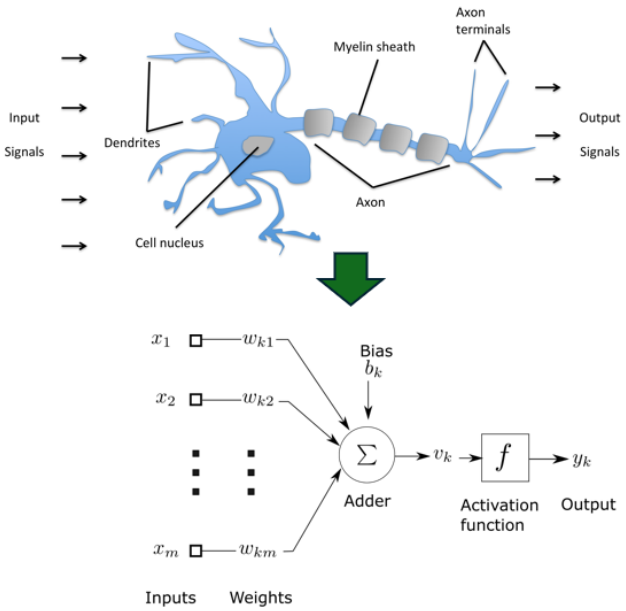


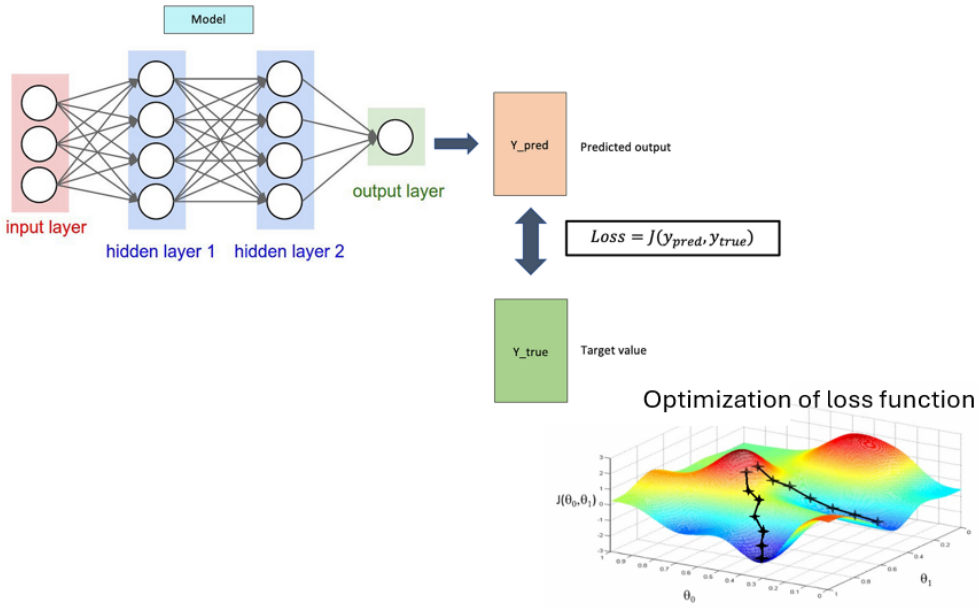


https://www.v7labs.com/blog/reinforcement-learning-applications

Subfield of AI: Deep Learning (DL)

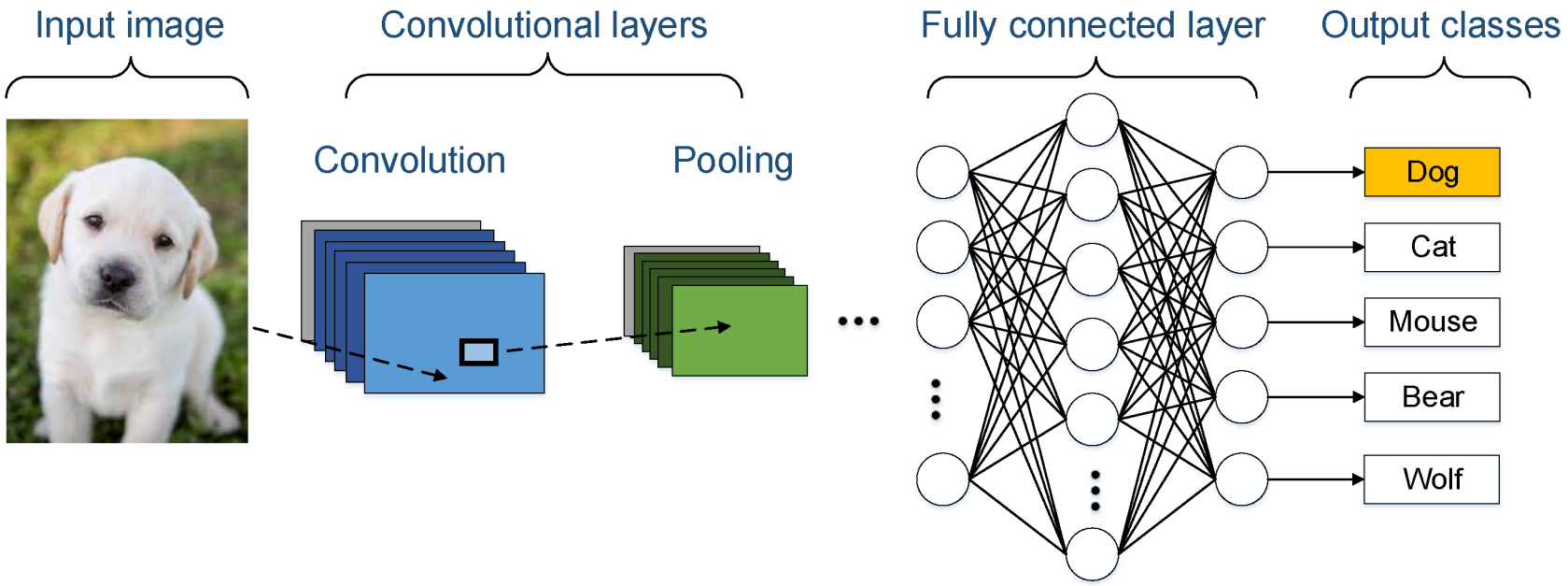
Deep Learning (DL) is a type of machine learning that uses **artificial neural networks** to learn from and make decisions based on data.





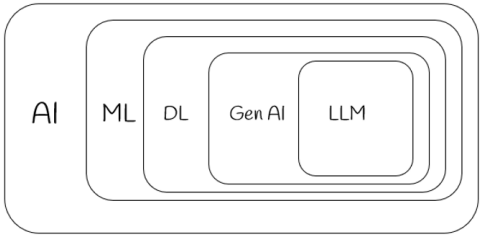
Subfield of AI: Deep Learning (DL)

Deep Learning (DL) is a type of machine learning that uses **artificial neural networks** to learn from and make decisions based on data.



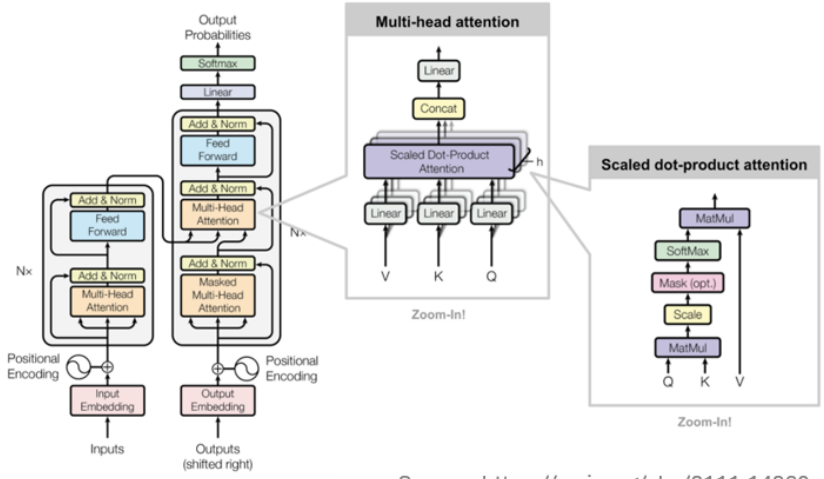
Subfield of AI : Generative AI (Gen AI)

Generative AI (Gen AI) refers to a class of artificial intelligence that focuses on creating new content or data that resembles existing data. This includes generating text, images, music, and more.

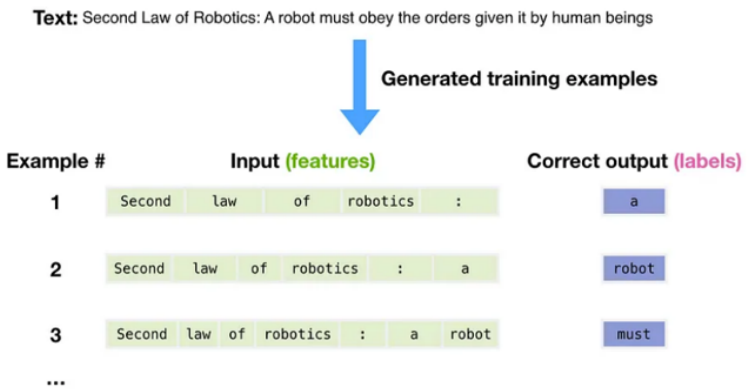


Subfield of AI : Large Language Models (LLM)

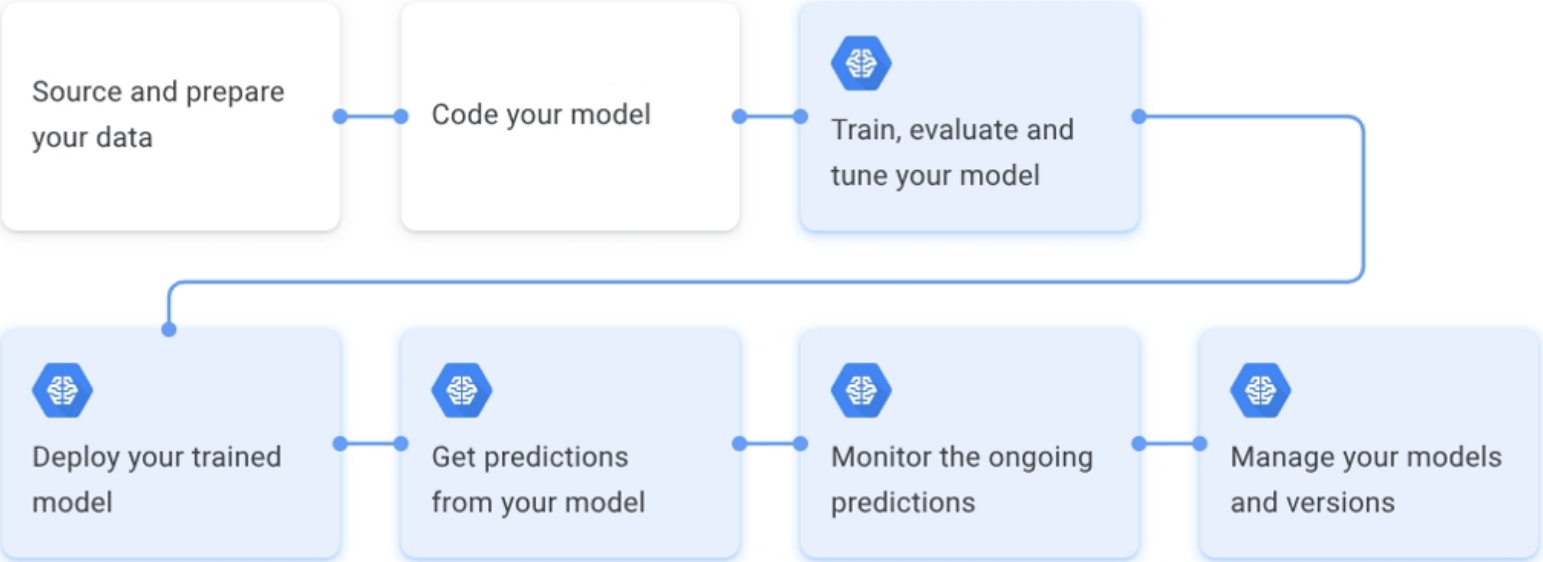
Large Language Models (LLM) are a specific type of generative AI designed to understand, generate, and manipulate human language. These models, often based on deep learning architectures like **transformers**, are trained on vast amounts of text data and can perform a wide range of tasks, including text generation, translation, summarization, and question-answering.



Source: https://arxiv.org/abs/2111.14260



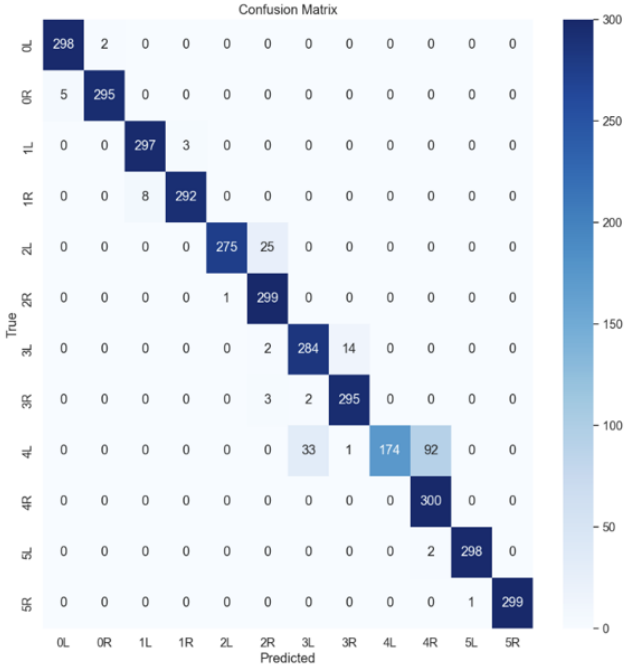
AI/ML Workflow



https://cloud.google.com/ai-platform/docs/ml-solutions-overview

Model Evaluation

Actual class \ Predicted class	P	N
P	True positives (TP)	False negatives (FN)
N	False positives (FP)	True negatives (TN)




```
Classification Report:
              precision    recall  f1-score   support

          0L       0.98      0.99      0.99       300
          0R       0.99      0.98      0.99       300
          1L       0.97      0.99      0.98       300
          1R       0.99      0.97      0.98       300
          2L       1.00      0.92      0.95       300
          2R       0.91      1.00      0.95       300
          3L       0.89      0.95      0.92       300
          3R       0.95      0.98      0.97       300
          4L       1.00      0.58      0.73       300
          4R       0.76      1.00      0.86       300
          5L       1.00      0.99      0.99       300
          5R       1.00      1.00      1.00       300

    accuracy                           0.95      3600
   macro avg       0.95      0.95      0.94      3600
weighted avg       0.95      0.95      0.94      3600
```

Outline

1 Intro. of AI

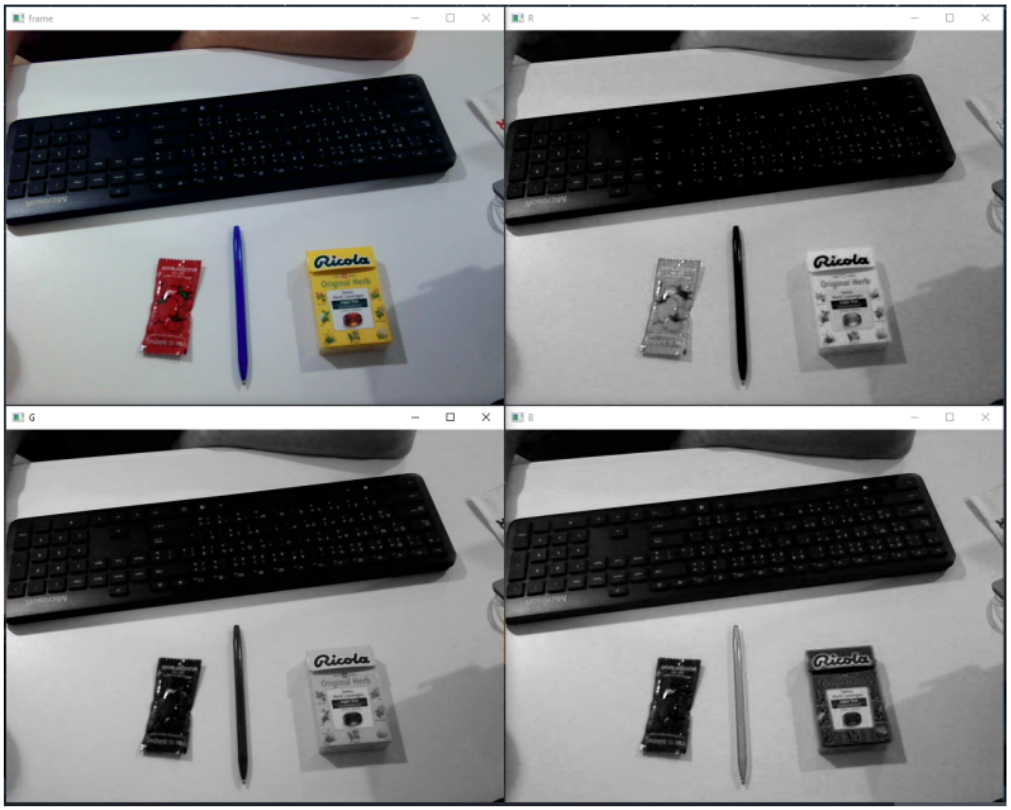
2 Images

3 CNN

4 Objection Detection

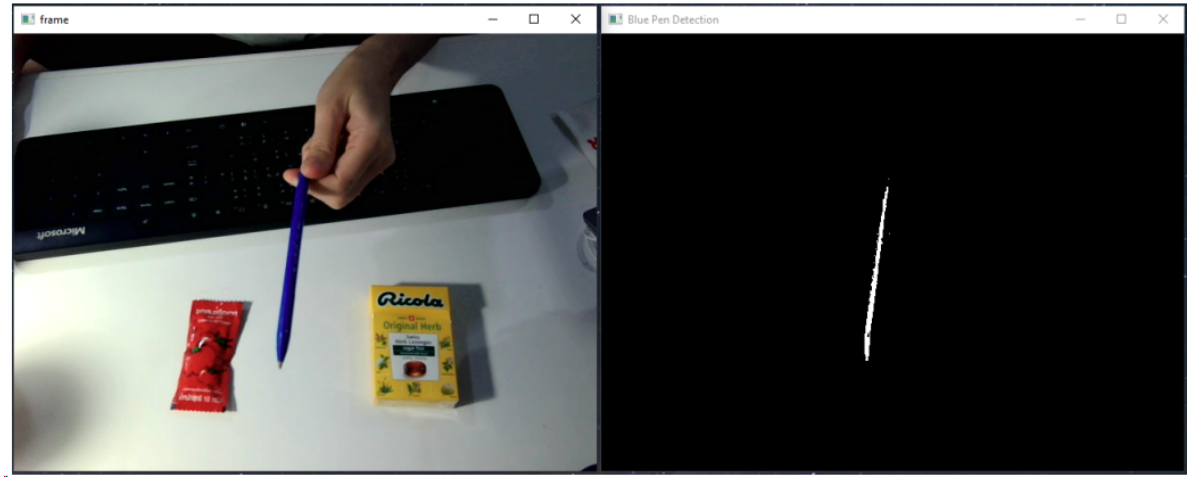
R G B

```
import cv2

#### Explore Color ####
cap = cv2.VideoCapture(0)
while True:
    _,frame = cap.read()
    print(frame.shape)

    #BGR
    R = frame[:,:,2]
    G = frame[:,:,1]
    B = frame[:,:,0]

    cv2.imshow("frame",frame)
    cv2.imshow("R",R)
    cv2.imshow("G",G)
    cv2.imshow("B",B)

    key = cv2.waitKey(1) #& 0xFF
    if key == ord('q'):
        break

cv2.destroyAllWindows()
```



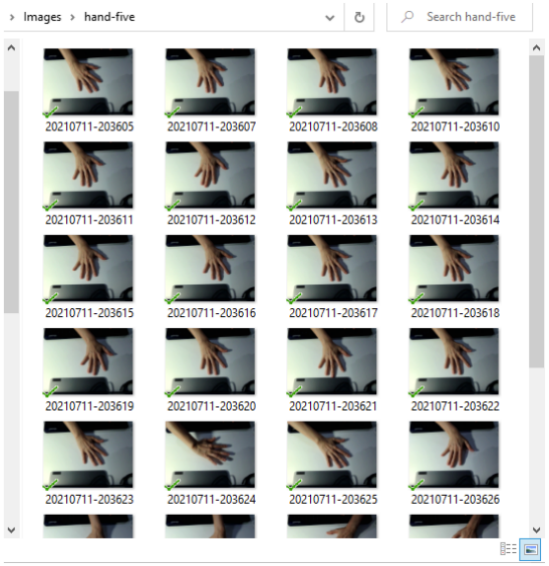
Color Threshold

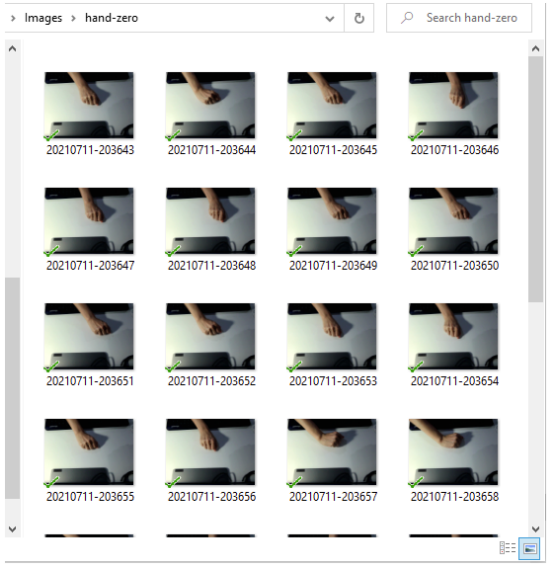
```
import cv2
### Object Detection with Color Threshold ###
cap = cv2.VideoCapture(0)
while True:
    _,frame = cap.read()
    print(frame.shape)

    #BGR
    R = frame[:,:,2]
    G = frame[:,:,1]
    B = frame[:,:,0]

    cv2.imshow("frame",frame)

    blue_pen = (R<90) & (G<90) & (B>120)
    blue_pen = 1.0*blue_pen
    cv2.imshow("Blue Pen Detection", blue_pen)

    key = cv2.waitKey(1) #& 0xFF
    if key == ord('q'):
        break

cv2.destroyAllWindows()
```



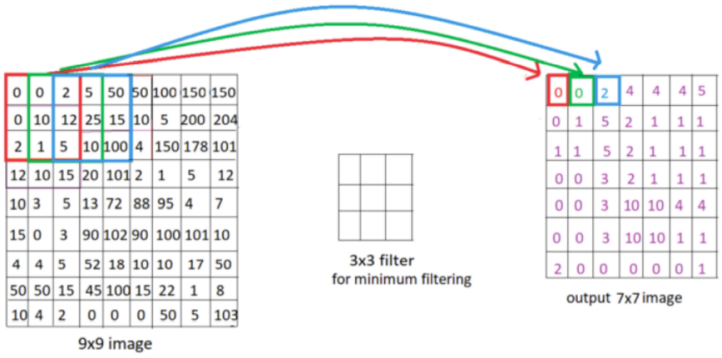
Collect Images

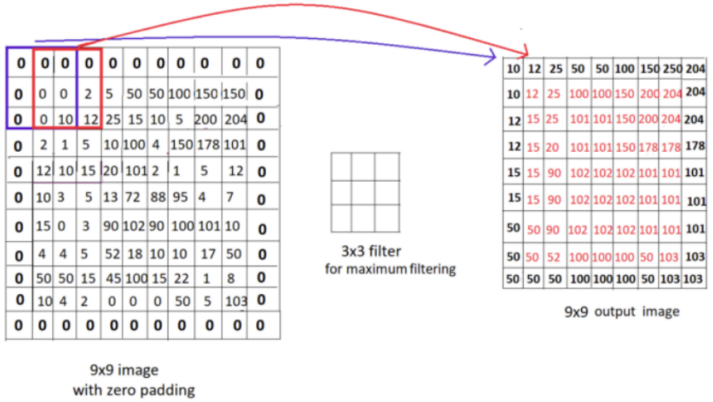
```python
import cv2
import time

#### Read From USB Camera ####
cap = cv2.VideoCapture(0)
while True:
    _,frame = cap.read()
    print(frame.shape)
    cv2.imshow("frame",frame)

    key = cv2.waitKey(1) #& 0xFF

    if key == ord('y'):
        timestr = time.strftime("%Y%m%d-%H%M%S")
        cv2.imwrite(filename="Images/hand-five/" + timestr + ".jpg", img =
    frame)

    if key == ord('n'):
        timestr = time.strftime("%Y%m%d-%H%M%S")
        cv2.imwrite(filename="Images/hand-zero/" + timestr + ".jpg", img =
    frame)

    if key == ord('q'):
        break

cv2.destroyAllWindows()
```

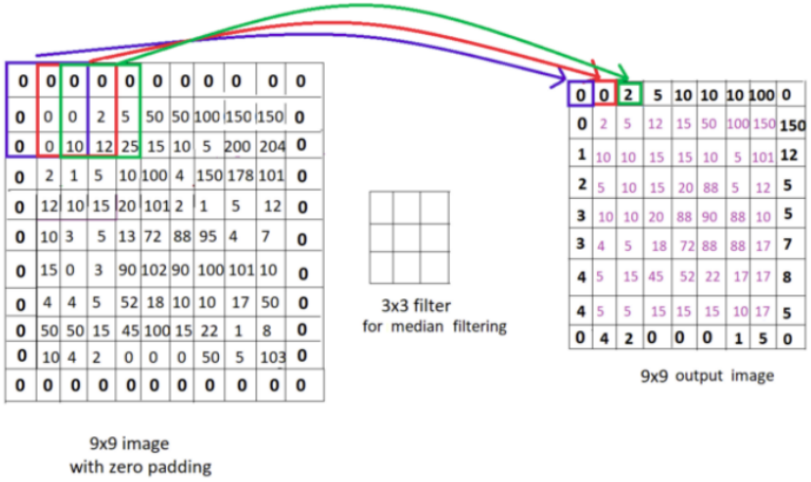




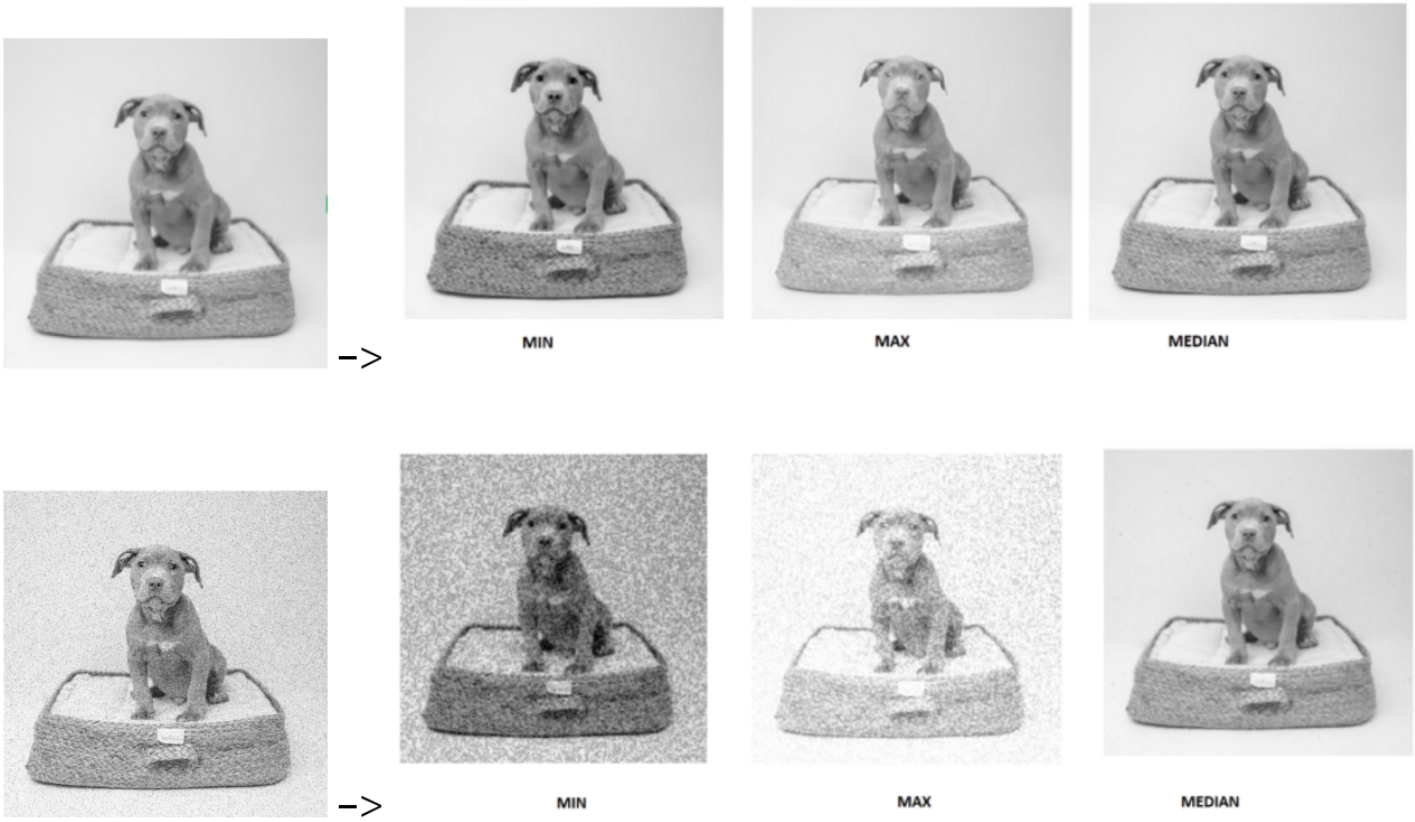
Filtering

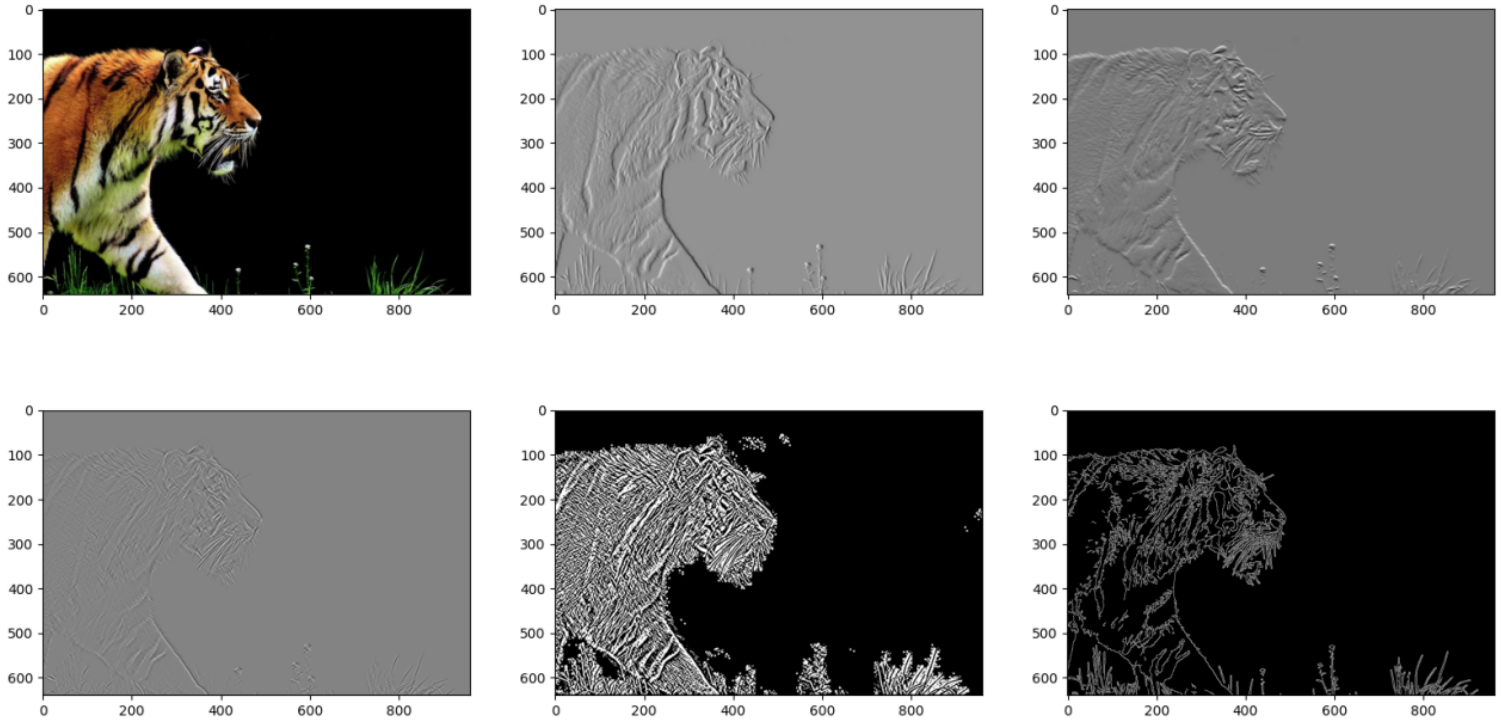

9x9 image
3x3 filter
for minimum filtering
output 7x7 image


9x9 image
with zero padding
3x3 filter
for maximum filtering
9x9 output image


9x9 image
with zero padding
3x3 filter
for median filtering
9x9 output image

Filtering




0
100
200
300
400
500
600
0
200
400
600
800

Outline

1 Intro. of AI

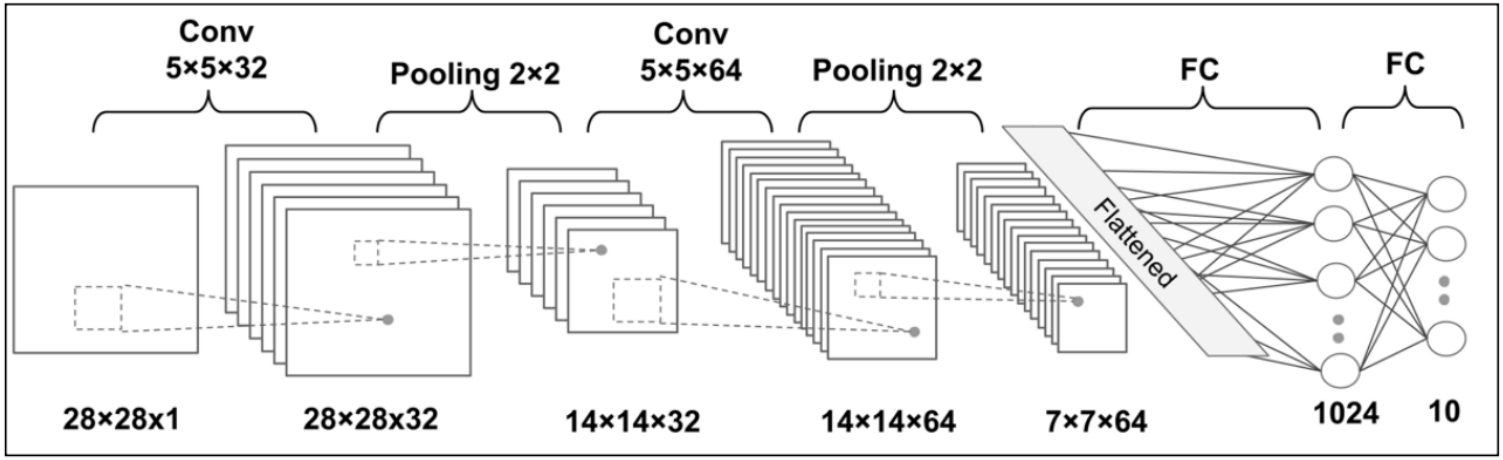
2 Images

3 CNN

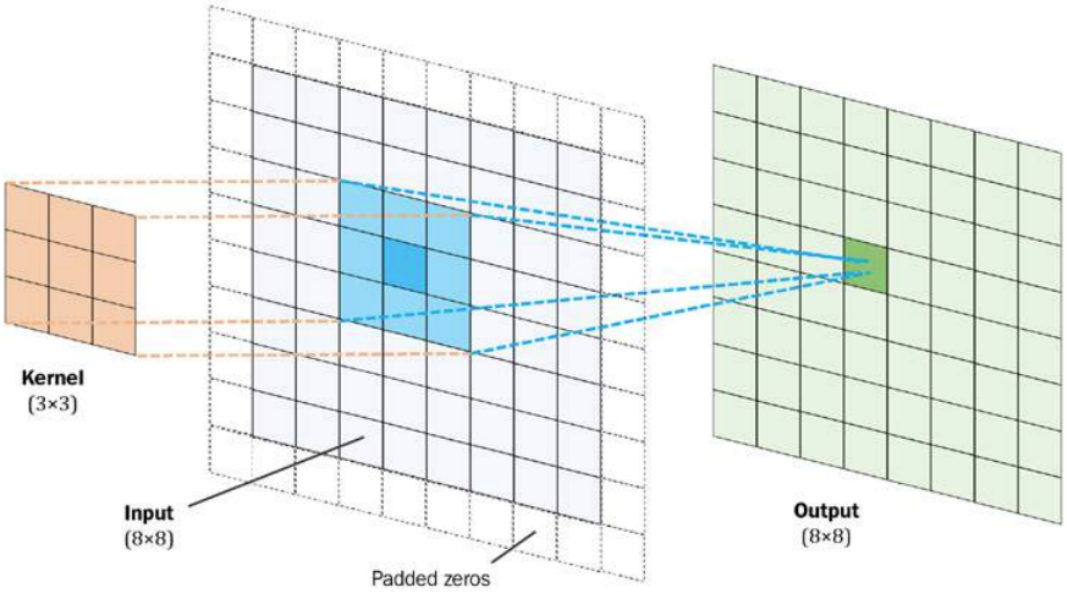
4 Objection Detection

CNN Architecture: A Layered Approach

- **Convolutional Layer**: Extracts features using filters. It applies a sliding window. This creates feature maps.
- **Pooling Layer**: Reduces dimensionality. It downsamples feature maps. Max pooling is a common technique.
- **Fully Connected Layer**: Performs high-level reasoning. It classifies the learned features. This is the final decision layer.



Convolutional Layer



What is Tensor ?

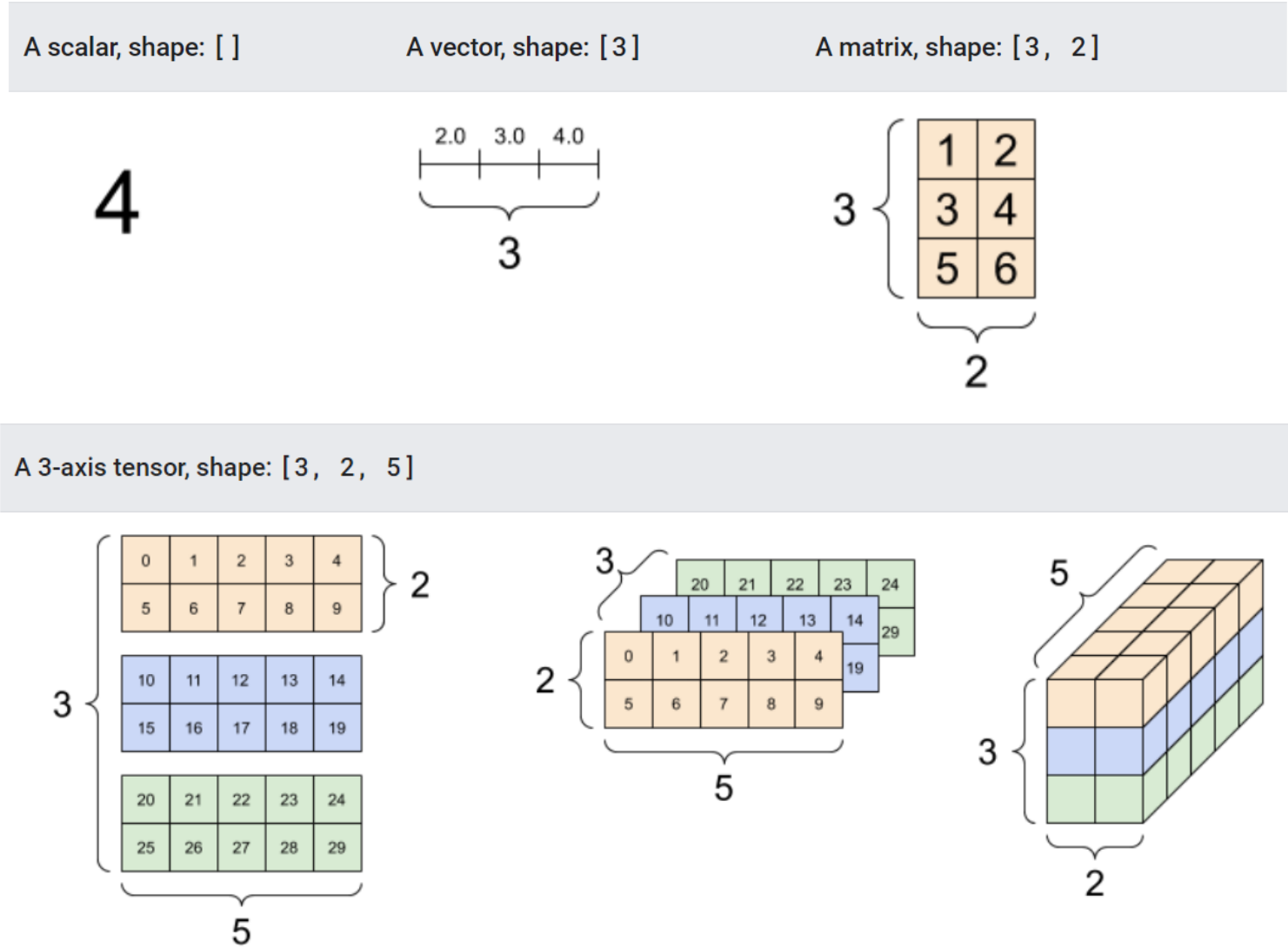
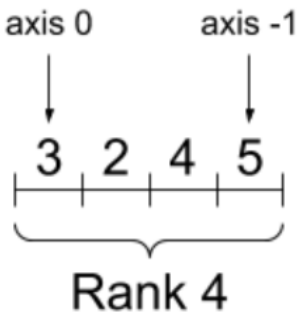


Image source: https://www.tensorflow.org/guide/tensor

What is Tensor ?

A rank-4 tensor, shape: [3, 2, 4, 5]



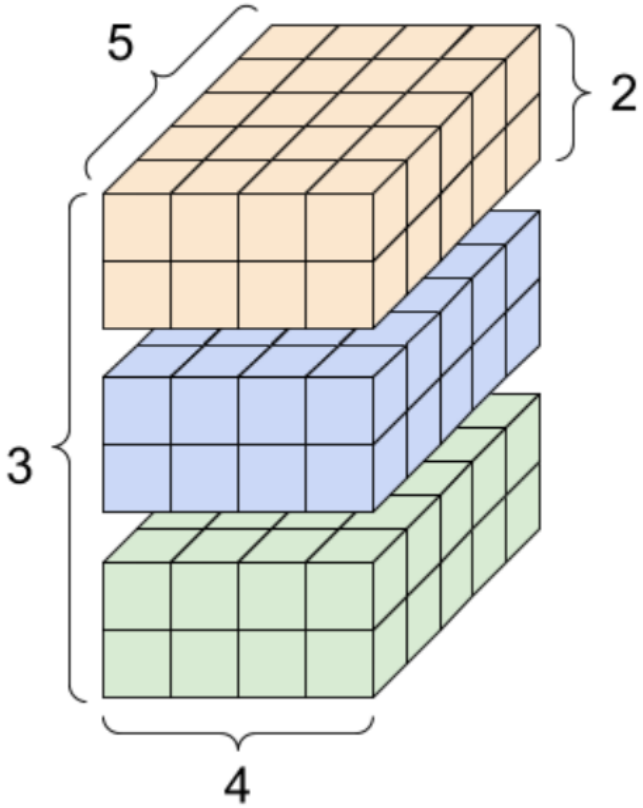
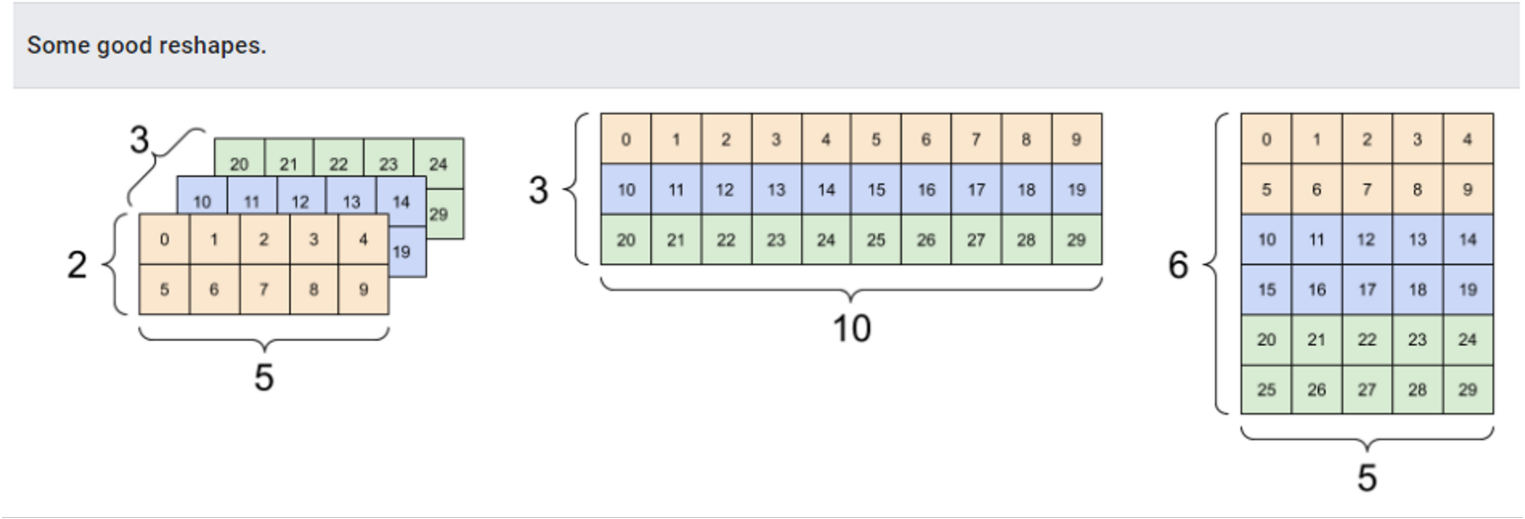


Image source: https://www.tensorflow.org/guide/tensor

Tensor: Reshape



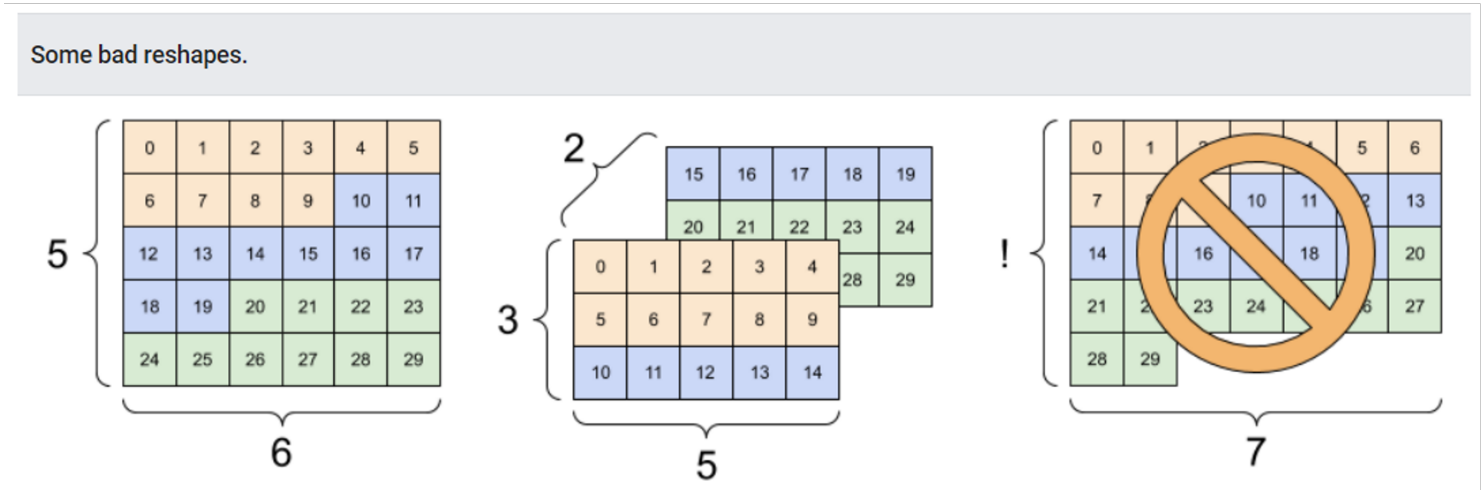


Image source: `https://www.tensorflow.org/guide/tensor`

Introduction to TensorFlow

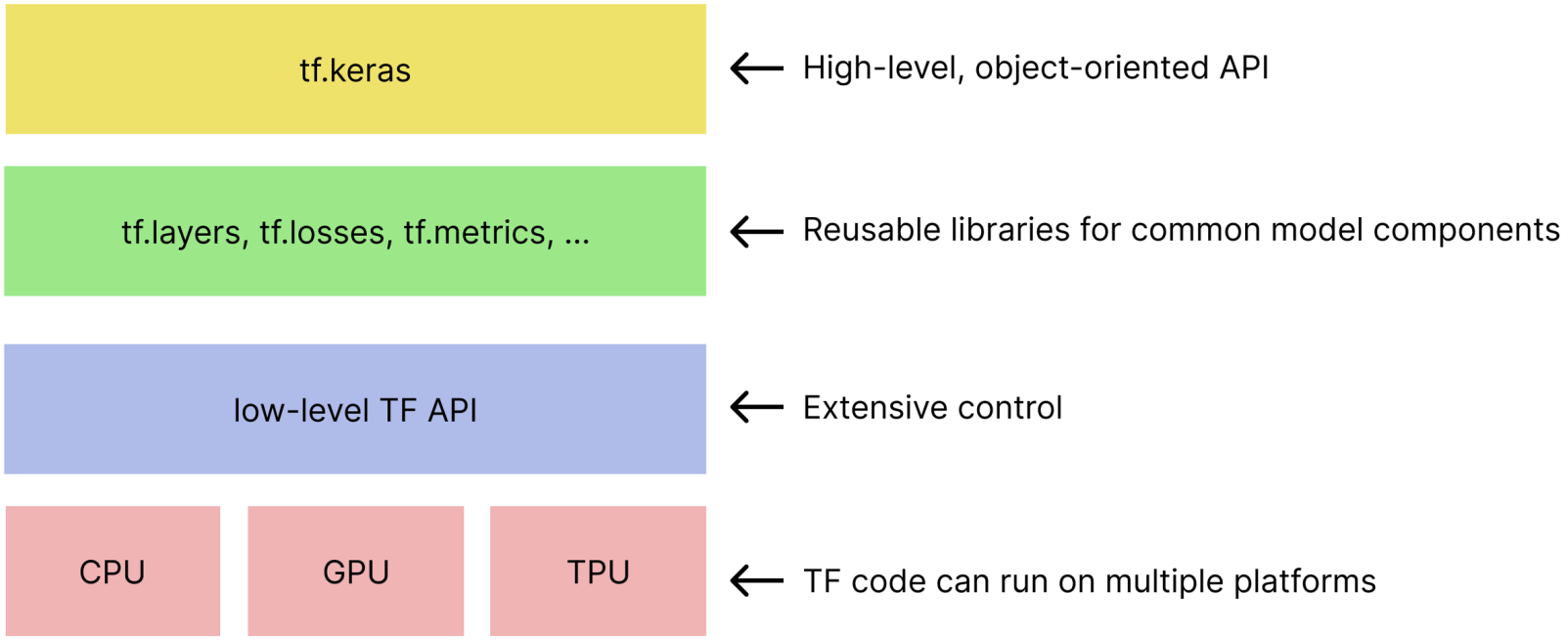
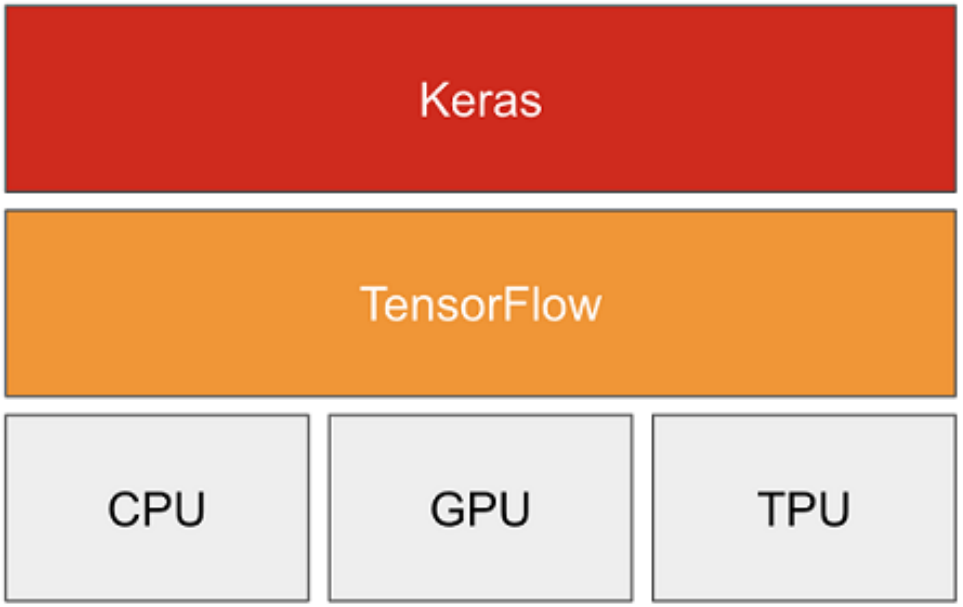


Image source: https://developers.google.com/machine-learning/crash-course/first-steps-with-tensorflow

TensorFlow & Keras



Deep learning development: layers, models, optimizers, losses, metrics...

Tensor manipulation infrastructure: tensors, variables, automatic differentiation, distribution...

Hardware: execution

Comparison: CPU vs GPU vs TPU

Feature	**CPU**	**GPU**	**TPU**
Designed for	General-purpose computing	Parallel processing (graphics, ML)	Machine Learning (mainly TensorFlow)
Core Count	Few (2–64 cores)	Hundreds to thousands	Matrix-focused units (MACs)
Memory	High latency, low bandwidth	High bandwidth VRAM	Very high bandwidth HBM
Efficiency	Flexible, general use	Good at large parallel tasks	Extremely efficient for ML ops
Best for	Operating systems, apps	Deep learning, rendering	Deep learning training / inference
Example	Intel i9, AMD Ryzen	NVIDIA T4, A100	Google TPU v2, v3, v4

Keras

https://keras.io/

keras.io

Keras GET STARTED GUIDES API EXAMPLES KERAS TUNER KERAS RS KERAS HUB SEARCH

Developer Guides

VIEW ALL

```
inputs = keras.Input(shape=(28, 28, 1))
x = inputs
x = layers.Conv2D(16, 3, activation="relu")(x)
x = layers.Conv2D(32, 3, activation="relu")(x)
x = layers.MaxPooling2D(3)(x)
x = layers.Conv2D(32, 3, activation="relu")(x)
x = layers.Conv2D(16, 3, activation="relu")(x)
x = layers.GlobalMaxPooling2D()(x)
x = layers.Dropout(0.5)
outputs = layers.Dense(10)

model = keras.Model(inputs, outputs)
model.summary()
```

The Functional API

Starting from the beginning and learn how to build models using the functional building pattern.

VIEW GUIDE

```
model.compile(
    optimizer="rmsprop",
    loss="categorical_crossentropy",
    metrics=["accuracy"],
)

history = model.fit(
    x_train,
    y_train,
    batch_size=64,
    epochs=2,
    validation_data=(x_val, y_val),
)
```

Training & evaluation with the built-in methods

Train and evaluate your model using model.fit(...).

VIEW GUIDE

```
class MLPBlock(keras.layers.Layer):
    def __init__(self):
        super().__init__()
        self.dense_1 = layers.Dense(32)
        self.dense_2 = layers.Dense(32)
        self.dense_3 = layers.Dense(1)

    def call(self, inputs):
        x = self.dense_1(inputs)
        x = keras.activations.relu(x)
        x = self.dense_2(x)
        x = keras.activations.relu(x)
        return self.dense_3(x)
```

Making new layers and models via subclassing

Learn how to customize your model via subclassing Keras layers.

VIEW GUIDE

Activation Functions

A list of common activation functions used in Keras:

Activation Function	**String**
Rectified Linear Unit (ReLU)	`'relu'`
Sigmoid	`'sigmoid'`
Hyperbolic Tangent (tanh)	`'tanh'`
Softmax	`'softmax'`
Linear	`'linear'`
Exponential Linear Unit (ELU)	`'elu'`
Parametric Rectified Linear Unit (PReLU)	`'prelu'`
Rectified Exponential Linear Unit (ReLU-6)	`'relu6'`
Scaled Exponential Linear Unit (SELU)	`'selu'`
Swish	`'swish'`
Softplus	`'softplus'`

https://www.tensorflow.org/api_docs/python/tf/keras/activations

Activation Functions

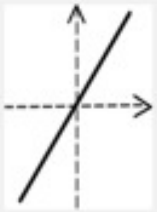
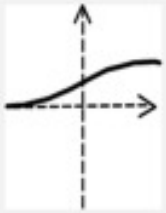
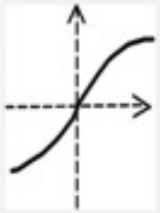
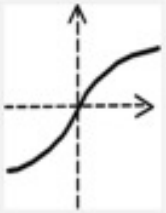
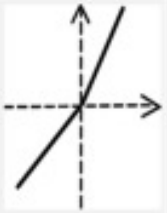
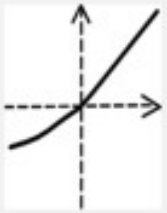
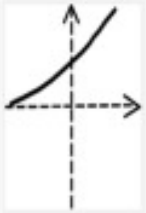
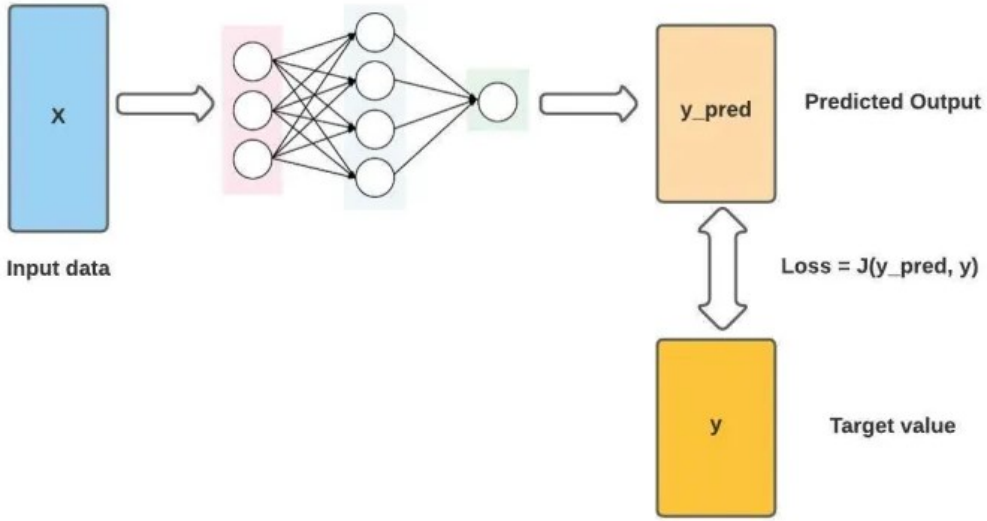
Activation Function									
	Identity	Binary Step	Logistic	TanH	ArcTan	ReLU	PreLU	ELU	SoftPlus
PLOT									
EQUATION	$f(x) = x$	$f(x) = \begin{cases} 0 \text{ for } x<0 \\ 1 \text{ for } x\geq 0 \end{cases}$	$f(x) = \frac{1}{1+e^{-x}}$	$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f(x) = tan^{-1}(x)$	$f(x) = \begin{cases} 0 \text{ for } x<0 \\ x \text{ for } x\geq 0 \end{cases}$	$f(x) = \begin{cases} \propto x \text{ for } x<0 \\ x \text{ for } x\geq 0 \end{cases}$	$f(x) = \begin{cases} \propto(e^{-x}-1) \text{ for } x<0 \\ x \text{ for } x\geq 0 \end{cases}$	$f(x) = log_e(1+e^x)$
DERIVATE	$f'(x) = 1$	$f'(x) = \begin{cases} 0 \text{ for } x\neq 0 \\ ? \text{ for } x=0 \end{cases}$	$f'(x) = f(x)(1-f(x))$	$f'(x) = 1-f(x)^2$	$f'(x) = \frac{1}{x^2+1}$	$f'(x) = \begin{cases} 0 \text{ for } x<0 \\ 1 \text{ for } x\geq 0 \end{cases}$	$f'(x) = \begin{cases} \propto \text{ for } x<0 \\ 1 \text{ for } x\geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x)+\propto \text{ for } x<0 \\ 1 \text{ for } x\geq 0 \end{cases}$	$f'(x) = \frac{1}{1+e^{-x}}$

Image Source: https://www.sciencedirect.com/topics/engineering/activation-function

Loss

In machine learning, Loss function is used to find error or deviation in the learning process.



Source: https://neptune.ai/blog/pytorch-loss-functions

Loss

Keras provides quite a few loss function in the losses module and they are as follows.

Loss Function	**Formula**
mean_squared_error	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
mean_absolute_error	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$
mean_absolute_percentage_error	$\frac{1}{n}\sum_{i=1}^{n}\left\lvert\frac{y_i-\hat{y}_i}{y_i}\right\rvert \times 100$
mean_squared_logarithmic_error	$\frac{1}{n}\sum_{i=1}^{n}(\log(1+y_i) - \log(1+\hat{y}_i))^2$
squared_hinge	$\max(0, 1 - y \cdot \hat{y})^2$
hinge	$\max(0, 1 - y \cdot \hat{y})$
categorical_hinge	$\sum_i \max(0, 1 - y_i \cdot \hat{y}_i)$
logcosh	$\log(\cosh(\text{predicted} - \text{actual}))$
huber_loss	$\begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{for } \lvert y-\hat{y}\rvert \leq \delta \\ \delta(\lvert y-\hat{y}\rvert - \frac{1}{2}\delta) & \text{otherwise} \end{cases}$
categorical_crossentropy	$-\sum_i y_i \cdot \log(\hat{y}_i)$
sparse_categorical_crossentropy	$-\sum_i \text{one_hot}(y_i) \cdot \log(\text{softmax}(\hat{y}_i))$
binary_crossentropy	$-\sum_i (y_i \cdot \log(\hat{y}_i) + (1-y_i) \cdot \log(1-\hat{y}_i))$
kullback_leibler_divergence	$\sum_i P(i) \cdot \log\left(\frac{P(i)}{Q(i)}\right)$
poisson	$\hat{y}_i - y_i \cdot \log(\hat{y}_i)$
cosine_proximity	$1 - \frac{\sum_i y_i \cdot \hat{y}_i}{\sqrt{\sum_i y_i^2 \cdot \sum_i \hat{y}_i^2}}$
is_categorical_crossentropy	$\begin{cases} -\log(\hat{y}_i) & \text{if } y_i = 1 \\ -\log(1-\hat{y}_i) & \text{if } y_i = 0 \end{cases}$

https://www.tensorflow.org/api_docs/python/tf/keras/losses

Optimizer

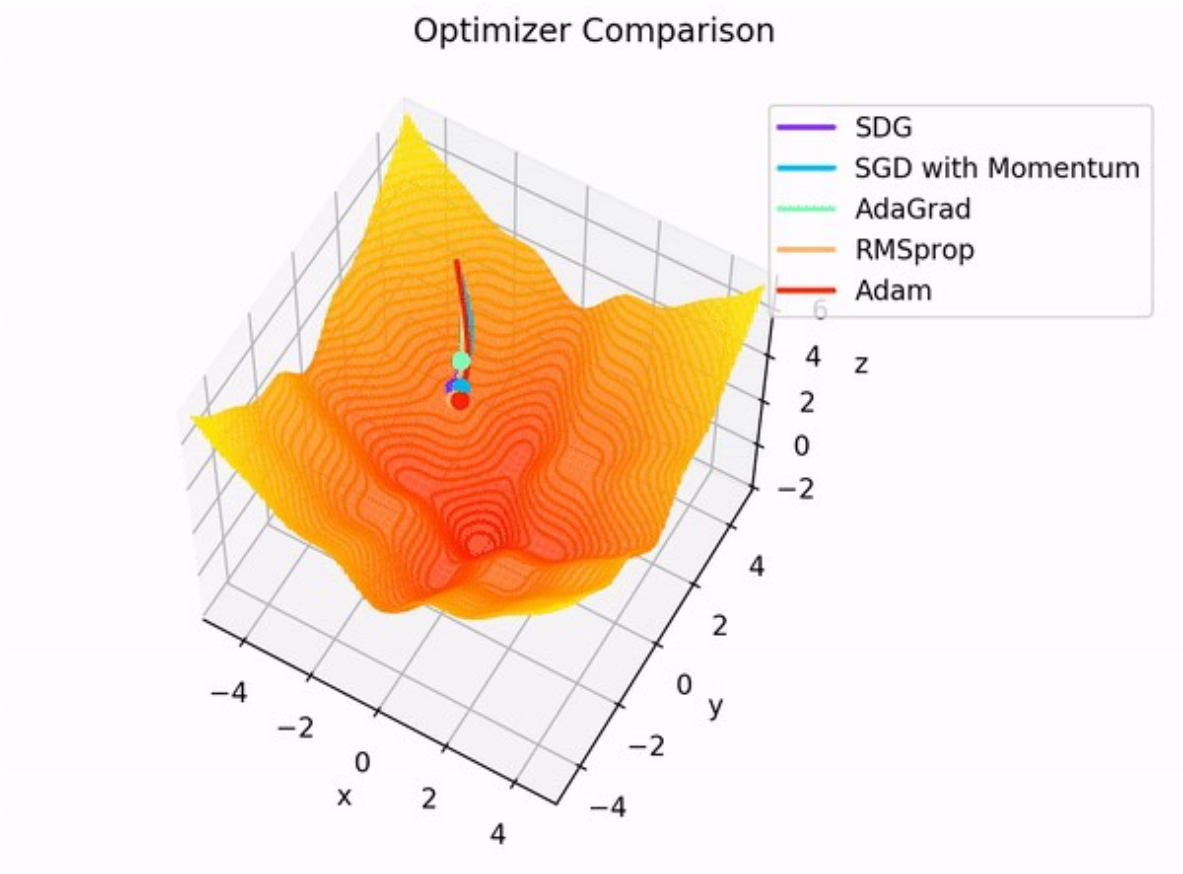


Image Source:

https://miro.medium.com/v2/resize:fit:1200/1*STiRp7PW5yIrvYZupZA6nw.gif

Optimizer

Optimizer	**String**
Stochastic Gradient Descent (SGD)	`'sgd'`
Adam	`'adam'`
RMSprop	`'rmsprop'`
Adagrad	`'adagrad'`
Adadelta	`'adadelta'`
Nadam	`'nadam'`
Adamax	`'adamax'`
FTRL	`'ftrl'`

https://www.tensorflow.org/api_docs/python/tf/keras/optimizers

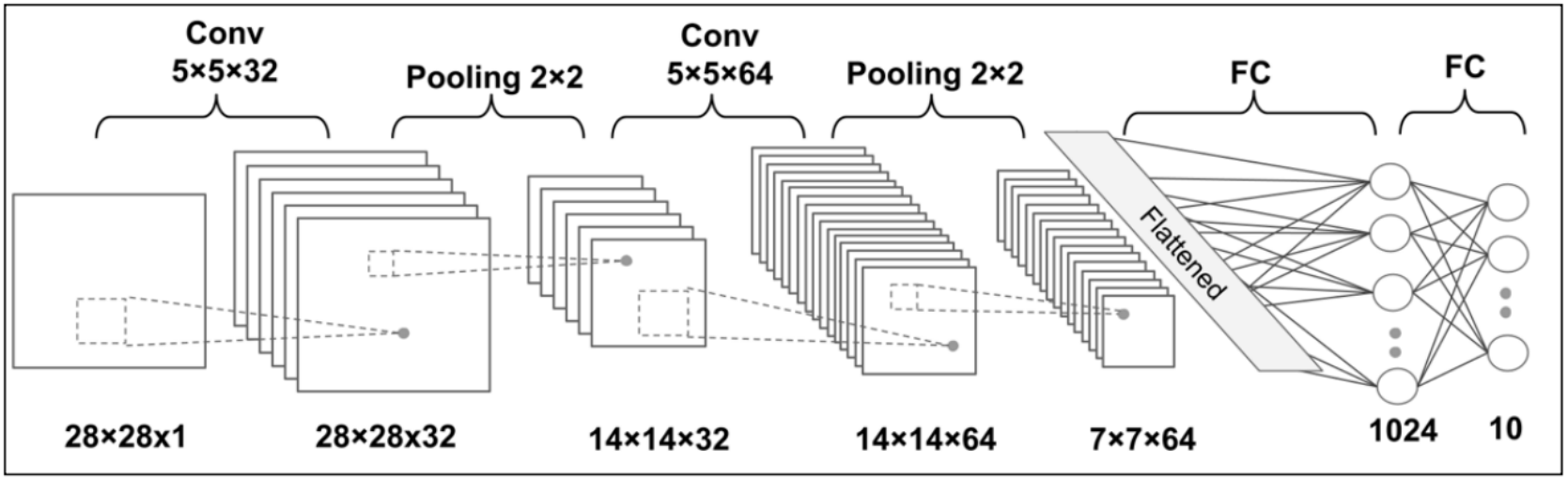
Metrics

In machine learning, Metrics is used to evaluate the performance of your model. It is similar to loss function, but not used in training process.

Metric	**String**
Accuracy	`'accuracy'`
Mean Squared Error (MSE)	`'mean_squared_error'`
Mean Absolute Error (MAE)	`'mean_absolute_error'`
Categorical Accuracy	`'categorical_accuracy'`
Binary Accuracy	`'binary_accuracy'`
Sparse Categorical Accuracy	`'sparse_categorical_accuracy'`
Top k Categorical Accuracy	`'top_k_categorical_accuracy'`
AUC (Area Under the Curve)	`'auc'`
Precision	`'precision'`
Recall	`'recall'`
F1 Score	`'f1'`

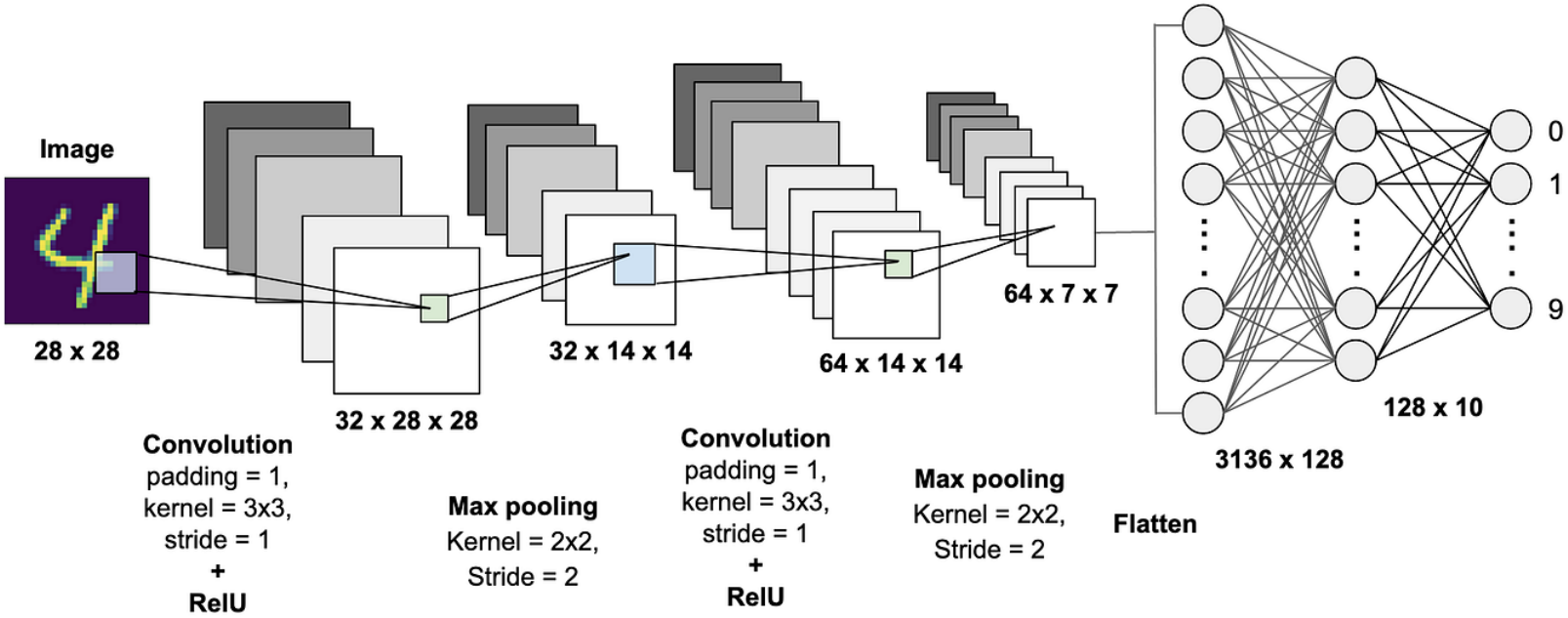
Example CNN

Create a sequential CNN model with the following layers



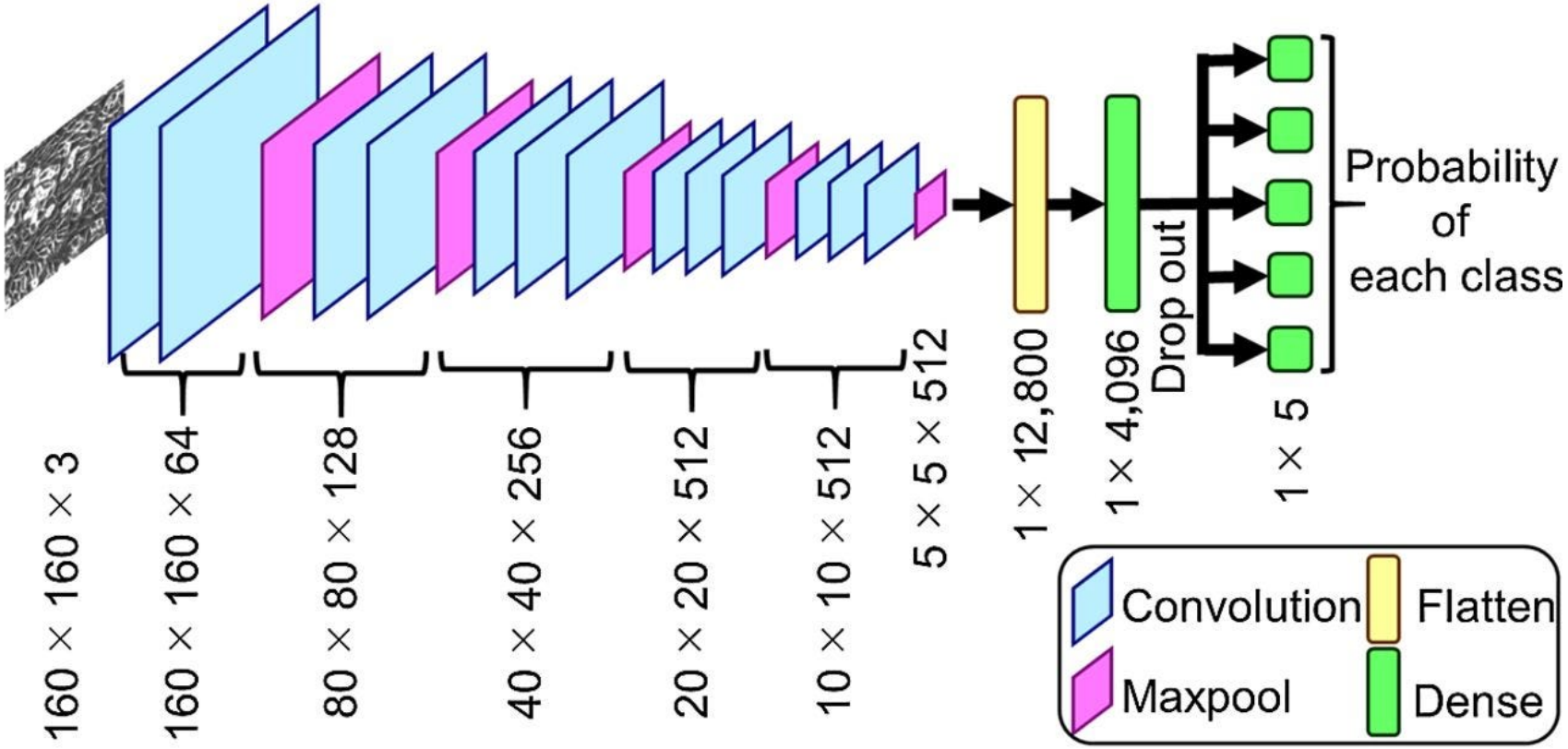
Example CNN

Create a sequential CNN model with the following layers



Example CNN

Create a sequential CNN model with the following layers



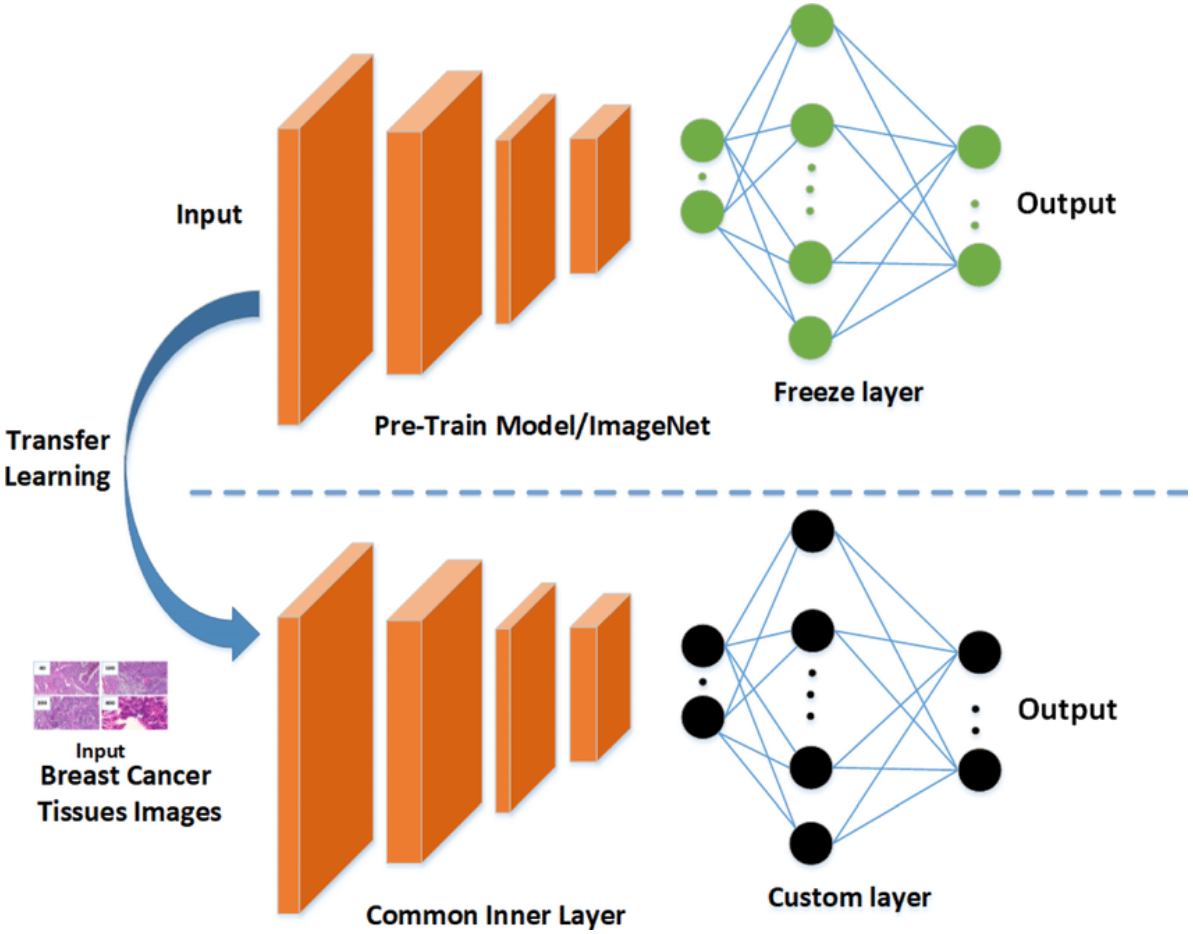
Pre-trained to Transfer Learning

Pre-training: Training a model on a large dataset (e.g., ImageNet, Wikipedia) to learn general features.

Transfer Learning: Using the knowledge learned from a pre-trained model and adapting it to a specific, often smaller, target task.

Example: Using a ResNet model pre-trained on ImageNet to classify X-ray images after fine-tuning on medical data.

Pre-trained to Transfer Learning



https://link.springer.com/article/10.1007/s00371-021-02153-y

Pre-trained to Transfer Learning

https://keras.io/api/applications/

keras.io/api/applications/

Keras GET STARTED GUIDES API EXAMPLES KERAS TUNER KERAS RS KERAS HUB

- Losses
- Data loading
- Built-in small datasets
- Keras Applications
  - Xception
  - EfficientNet B0 to B7
  - EfficientNetV2 B0 to B3 and S, M, L
  - ConvNeXt Tiny, Small, Base, Large, XLarge
  - VGG16 and VGG19
  - ResNet and ResNetV2
  - MobileNet, MobileNetV2, and MobileNetV3
  - DenseNet
  - NasNetLarge and NasNetMobile
  - InceptionV3
  - InceptionResNetV2
- Mixed precision
- Multi-device distribution

Available models

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
InceptionResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4
MobileNetV2	14	71.3%	90.1%	3.5M	105	25.9	3.8
DenseNet121	33	75.0%	92.3%	8.1M	242	77.1	5.4
DenseNet169	57	76.2%	93.2%	14.3M	338	96.4	6.3
DenseNet201	80	77.3%	93.6%	20.2M	402	127.2	6.7
NASNetMobile	23	74.4%	91.9%	5.3M	389	27.0	6.7
NASNetLarge	343	82.5%	96.0%	88.9M	533	344.5	20.0
EfficientNetB0	29	77.1%	93.3%	5.3M	132	46.0	4.9
EfficientNetB1	31	79.1%	94.4%	7.9M	186	60.2	5.6
EfficientNetB2	36	80.1%	94.9%	9.2M	186	80.8	6.5

Outline

1 Intro. of AI

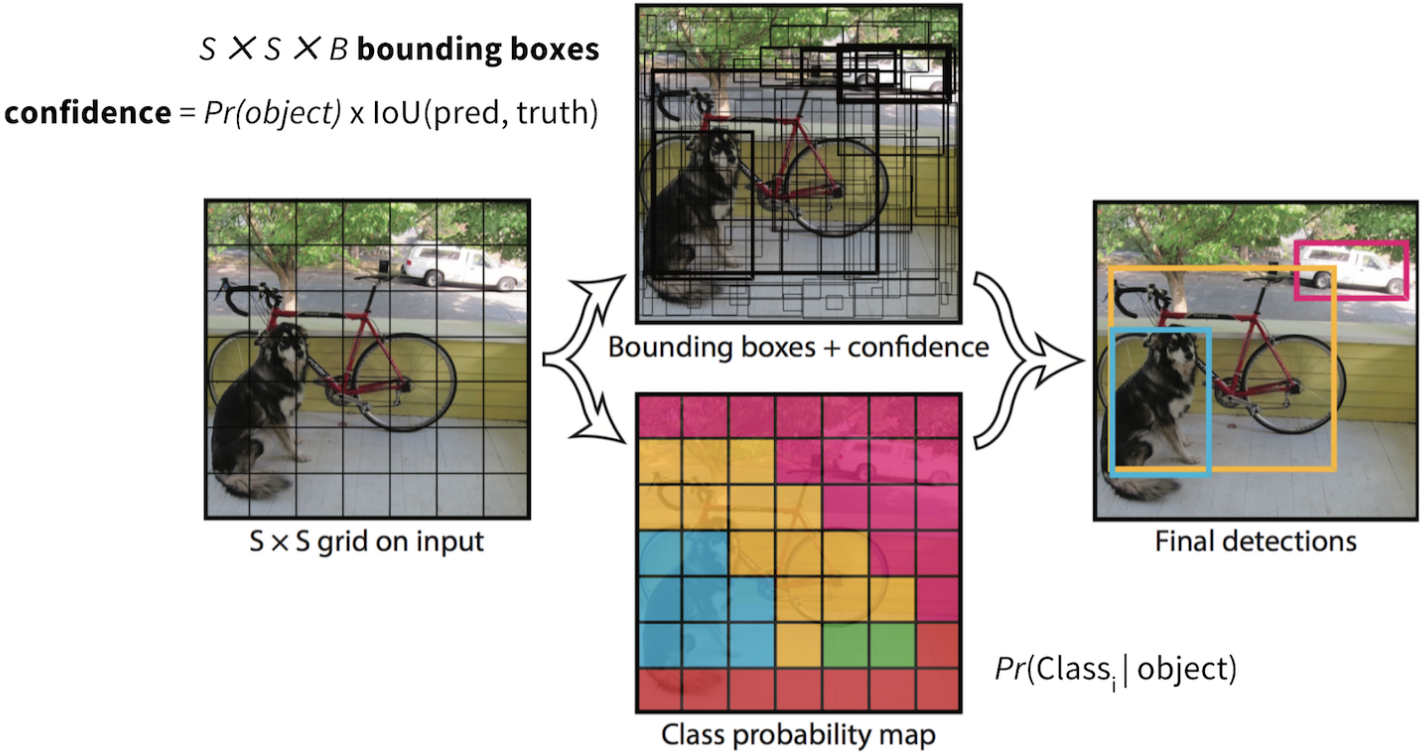
2 Images

3 CNN

4 Objection Detection

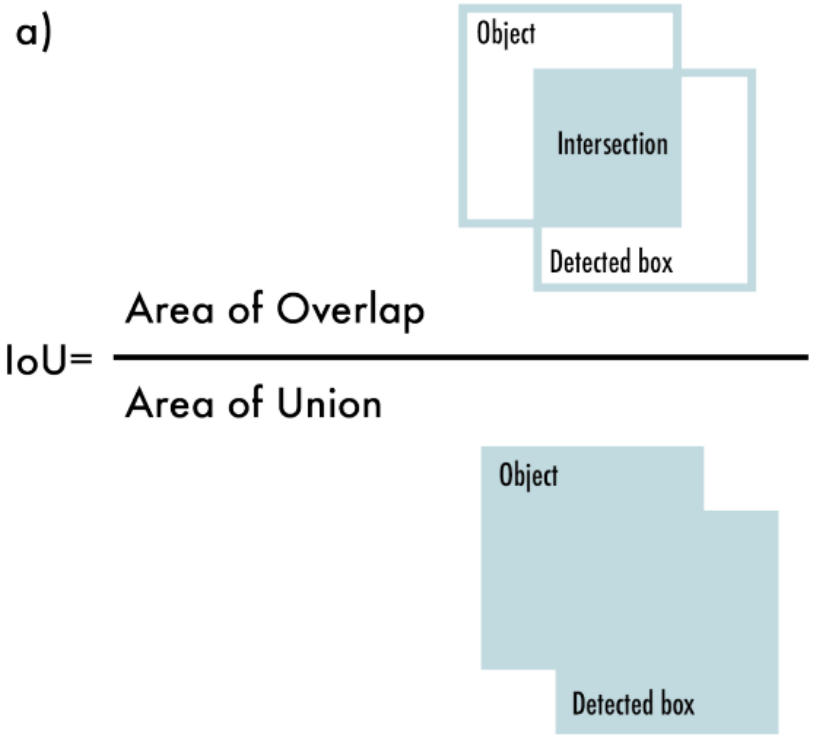
YOLO

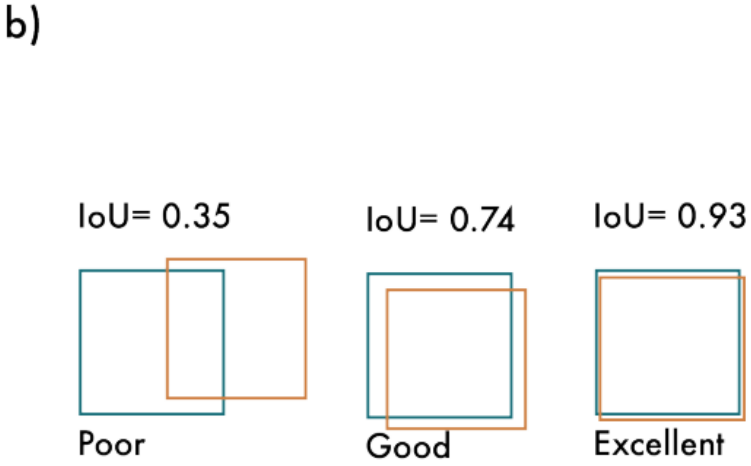
You Only Look Once: Unified, Real-Time Object Detection
https://arxiv.org/abs/1506.02640



$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

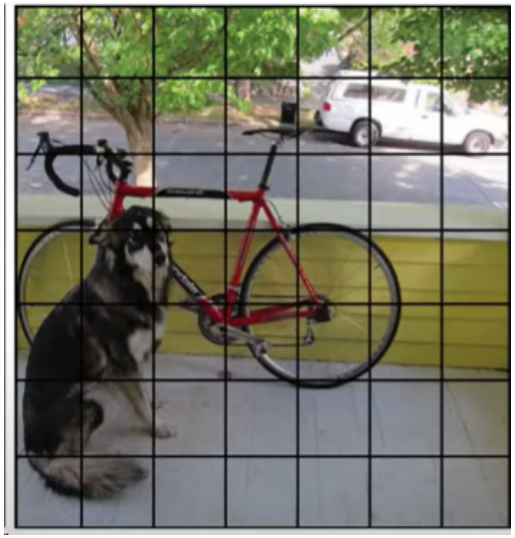
YOLO


a)
Object
Intersection
Detected box
IoU= Area of Overlap / Area of Union
Object
Detected box


b)
IoU= 0.35
IoU= 0.74
IoU= 0.93
Poor
Good
Excellent

YOLO

https://arxiv.org/abs/1506.02640

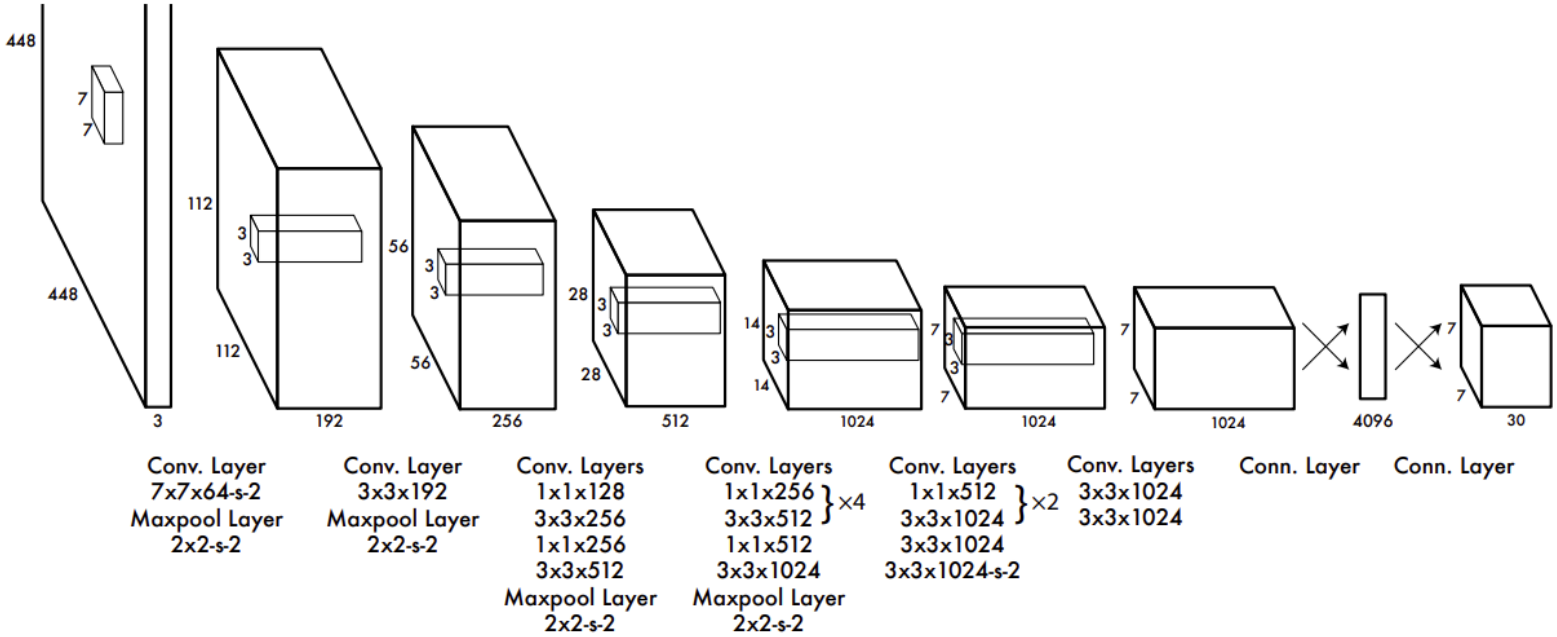


Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally we define confidence as $\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$. If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally the confidence prediction represents the IOU between the predicted box and any ground truth box.

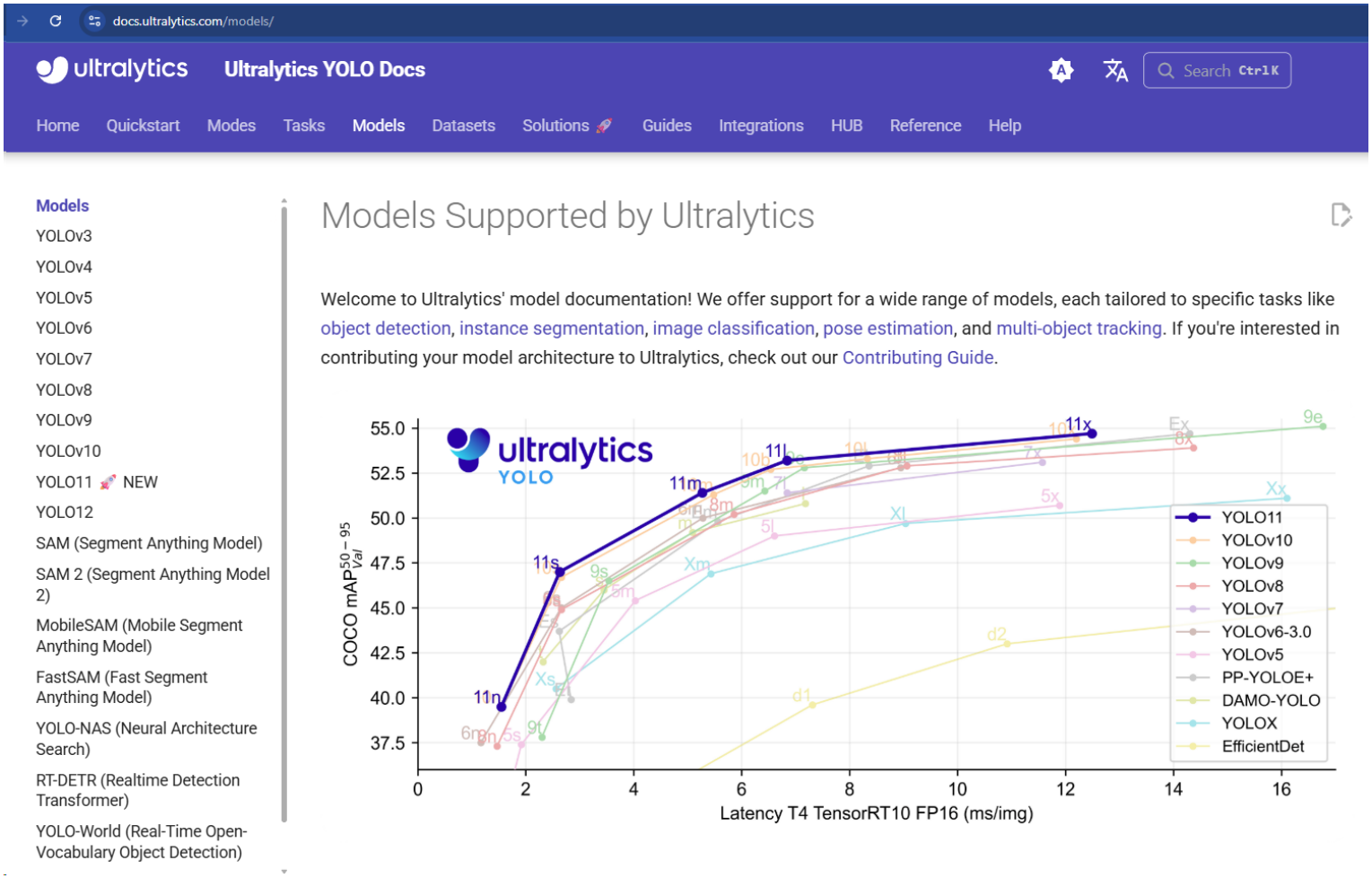
For evaluating YOLO on PASCAL VOC, we use S = 7, B = 2. PASCAL VOC has 20 labelled classes so C = 20. Our final prediction is a $7 \times 7 \times 30$ tensor

YOLO


448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2
Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x256
3x3x512
×4
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x512
3x3x1024
×2
3x3x1024
3x3x1024-s-2
Conv. Layers
3x3x1024
3x3x1024
Conn. Layer
Conn. Layer

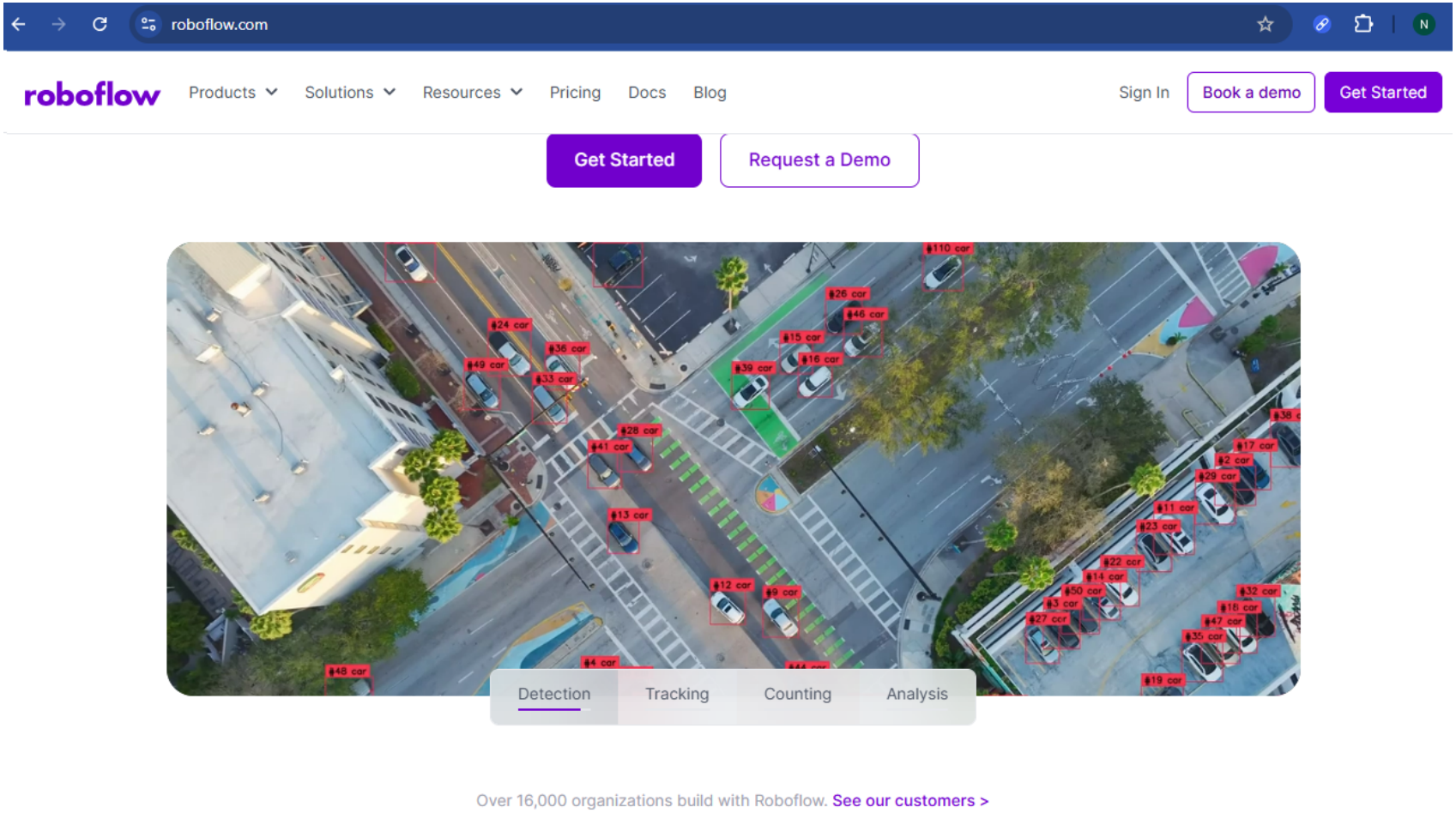
YOLO

https://docs.ultralytics.com/modes/



Roboflow

https://roboflow.com



Thank You!