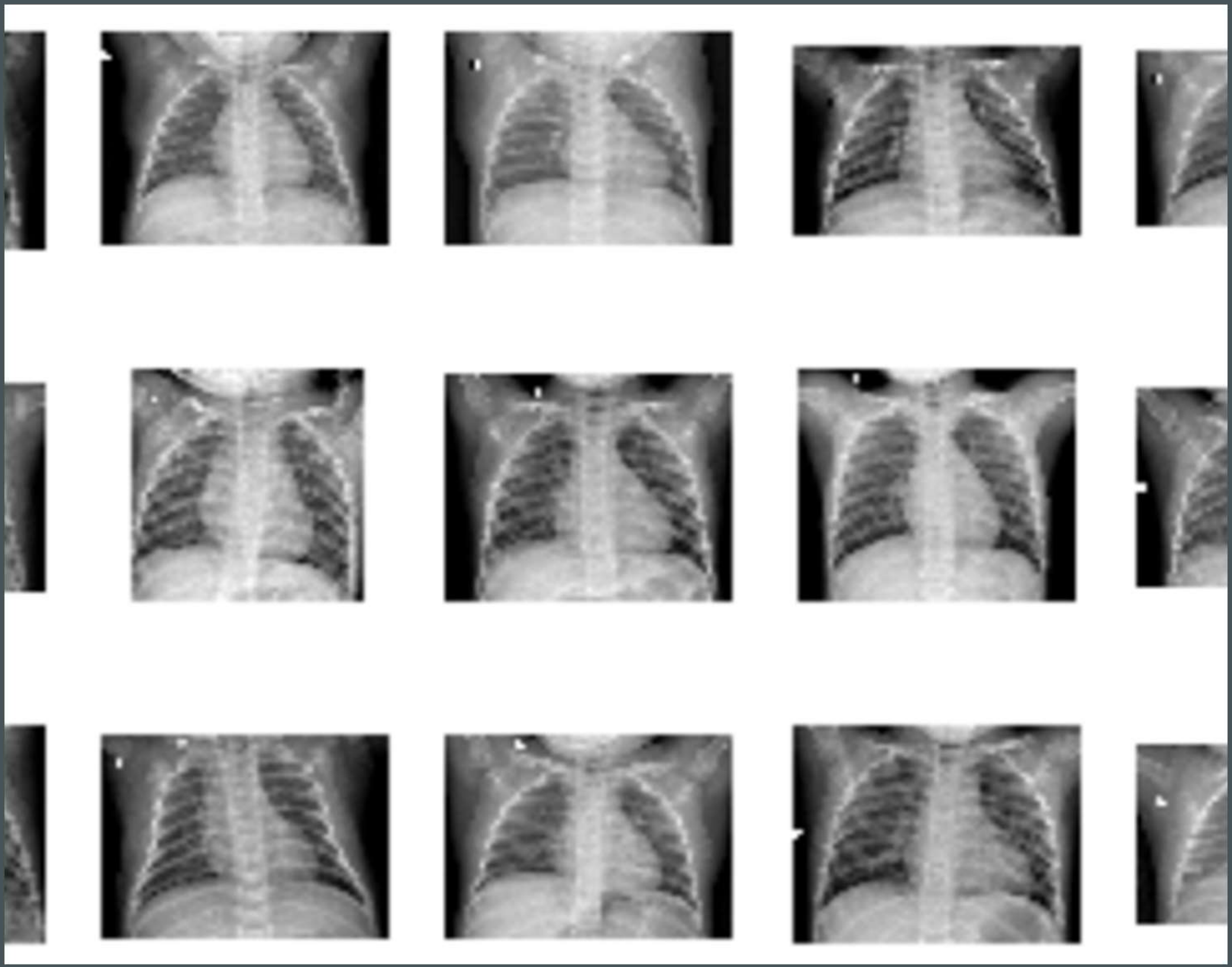
APPLYING CNN IN (PNEUMONIA) CHEST IMAGE CLASSIFICATION

TRAN LE



GOAL OF THE PROJECT

Learn CNN	Apply With Chest Image Problem
• Understand how CNN works (basically). • How to run basic CNN with Python	• Run a simple CNN • Tune hyperparameters and evaluate the optimal model

OUTLINE

1. Review about neural network	2. CNN	3. Apply CNN with the chest image problem.	4. A summarization about the result
		• A simple model • A model with callbacks • Tuning hyperparameter with and without callbacks	

1. NEURAL NETWORK OVERVIEW

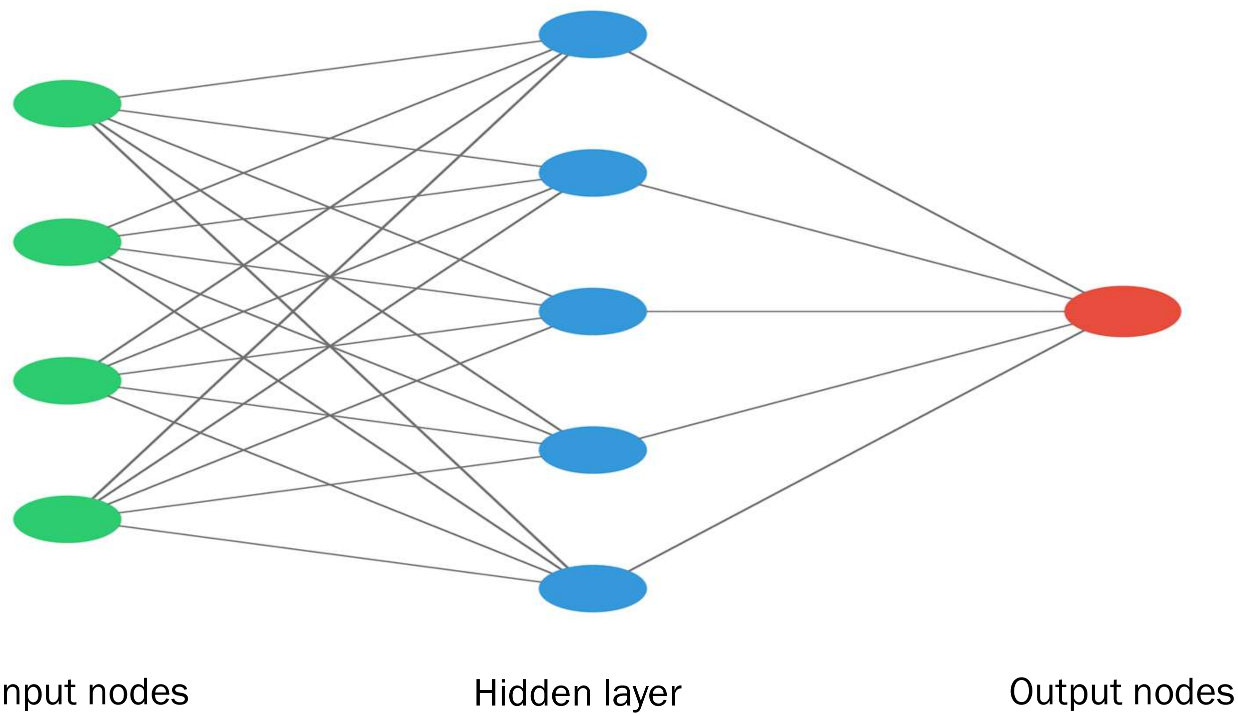
1. NEURAL NETWORK

a) Architecture

b) Connection between nodes: weights, bias, activation function,

c) How to find weights and bias: Backpropagation

a) Architecture



There could be one or more than one hidden layers

1. NEURAL NETWORK

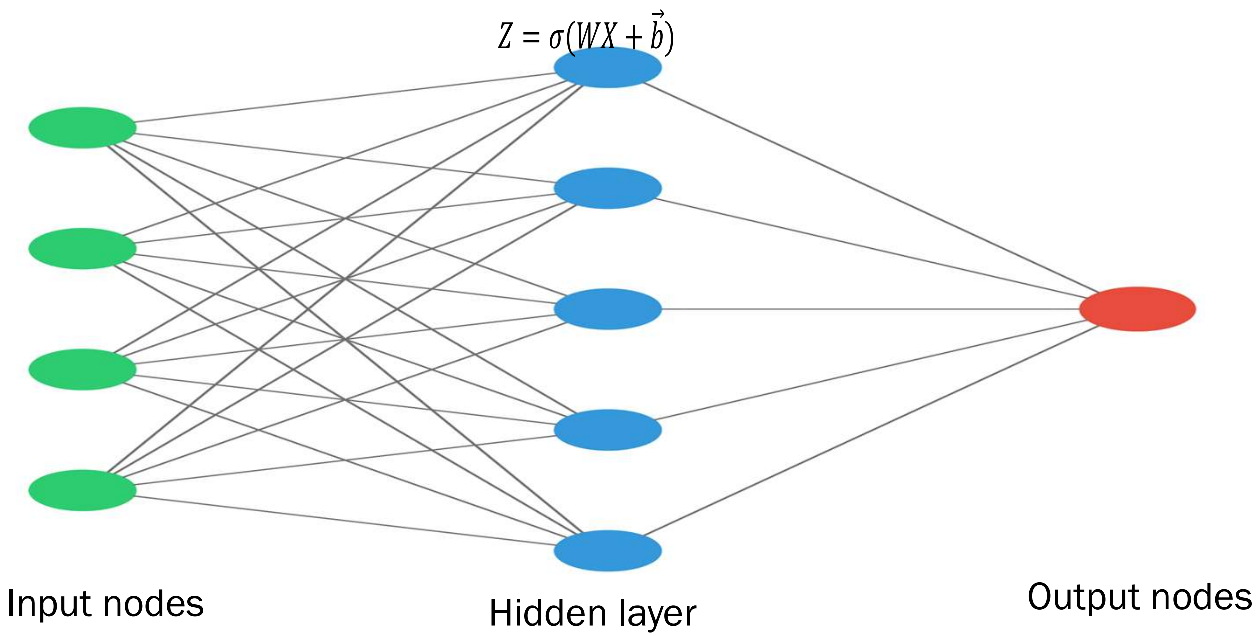
a) Architecture

b) Connection between nodes: weights, bias, activation function,

c) How to find weights and bias: Backpropagation

Input nodes

b) Connection between nodes: weights, bias, activation function.



W: weight

$\vec{b}$: bias

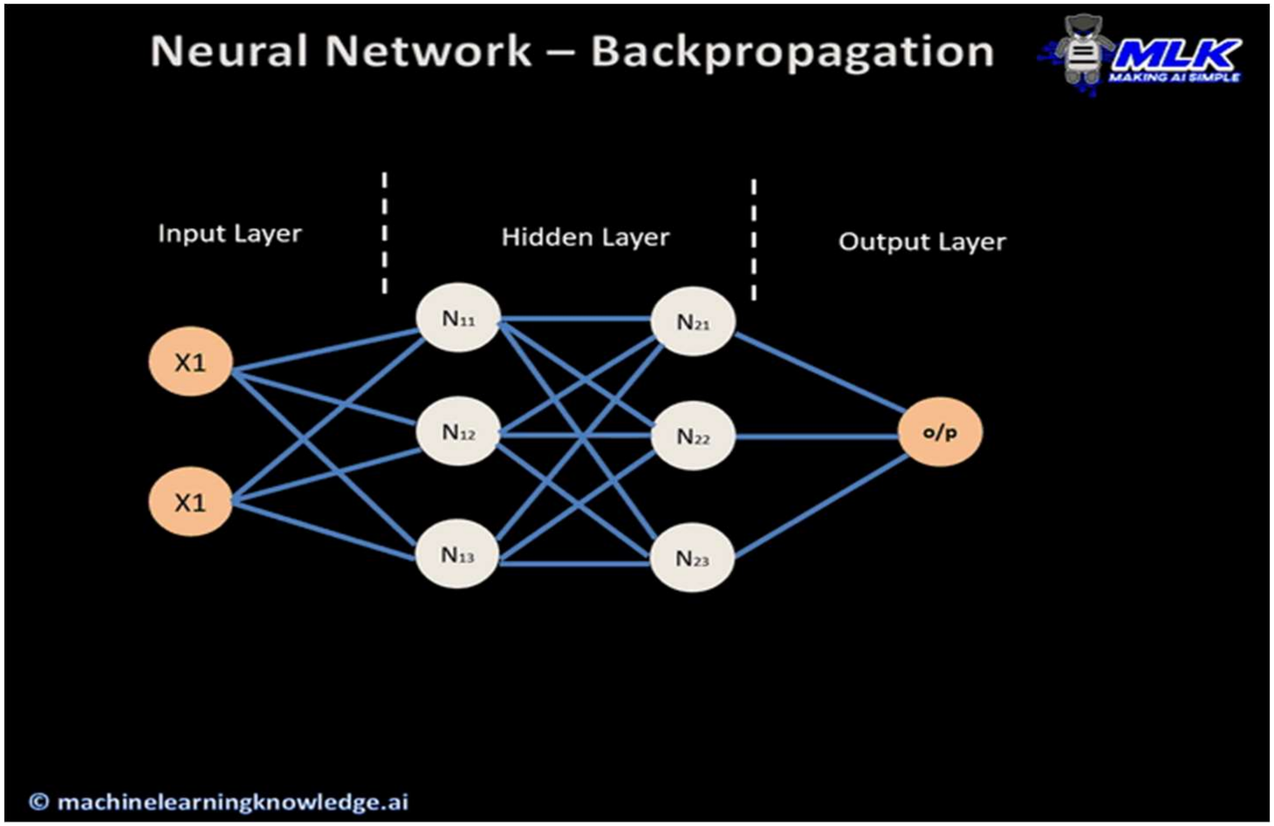
σ:activation function (Linear, RELU, Sigmoid, Solfmax, ...). Depending on type of problem to decide the adequate activation function.

1. NEURAL NETWORK

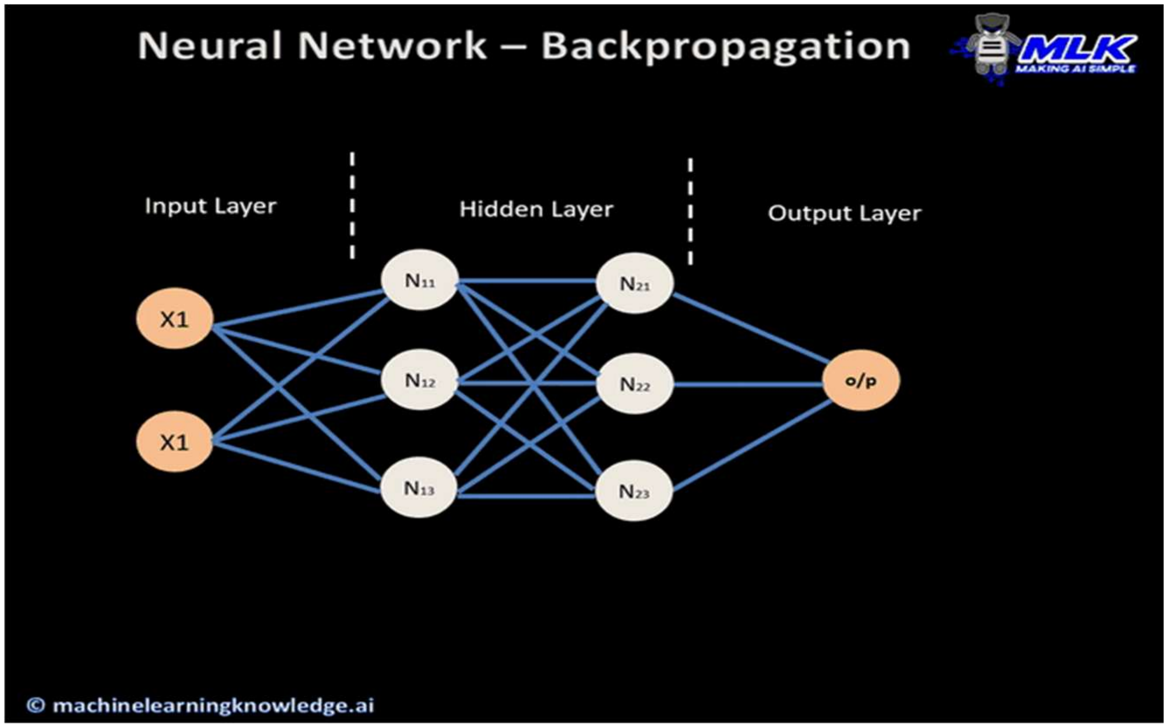
a) Architecture
b) Connection between nodes: weights, bias, activation function,
c) How to find weights and bias: Backpropagation

Input nodes

c) How to find weights and bias: Backpropagation



HOW TO FIND THE WEIGHTS AND BIAS



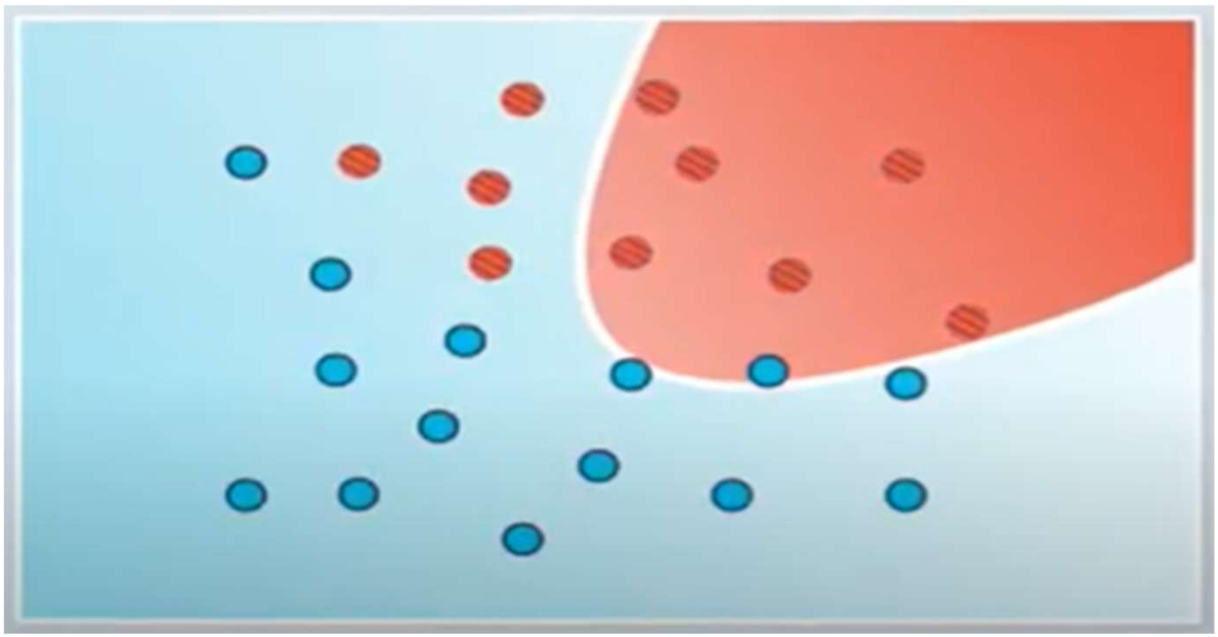
Epoch: The number of **epochs** is a hyperparameter that defines the number **times** that the learning algorithm will work through the entire training dataset.

Batch: The batch size is a hyperparameter that defines the **number of samples** to work through before updating the internal model parameters.

•**Batch Gradient Descent**. Batch Size = Size of Training Set
•**Stochastic Gradient Descent**. Batch Size = 1
•**Mini-Batch Gradient Descent**. 1 < Batch Size < Size of Training Set

Source: https://www.google.com/imgres?imgurl=https%3A%2F%2Fmachinelearningknowledge.ai%2Fwp-content%2Fuploads%2F2019%2F10%2FBackpropagation.gif&imgrefurl=https%3A%2F%2Fmachinelearningknowledge.ai%2Fanimated-explanation-of-feed-forward-neural-network-architecture%2F&tbnid=PrPkgqLRXSgkyM&vet=12ahUKEwjF082R1JPxAhXqgE4HHW7LChgQMygEegUIARDnAQ..i&docid=sRMKI6l9zyTc3M&w=640&h=480&q=backpropagation%20gif%20&ved=2ahUKEwjF082R1JPxAhXqgE4HHW7LChgQMygEegUIARDnAQ

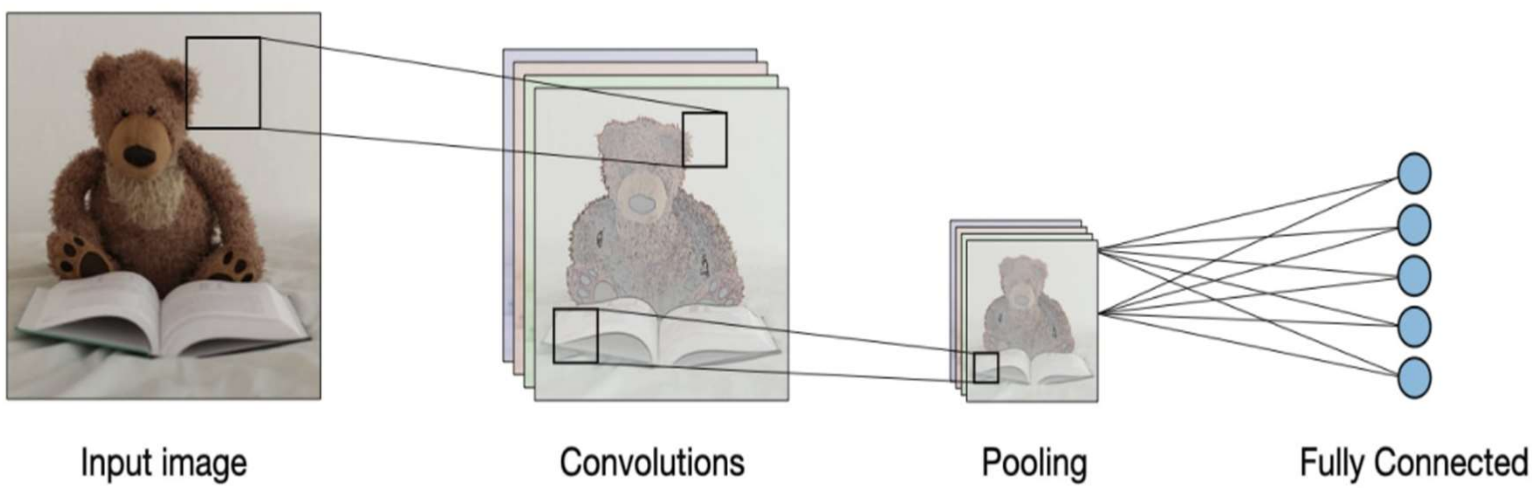
BATCH



Source: https://www.youtube.com/watch?v=cRd3q4BeRmg&ab_channel=AIQCAR

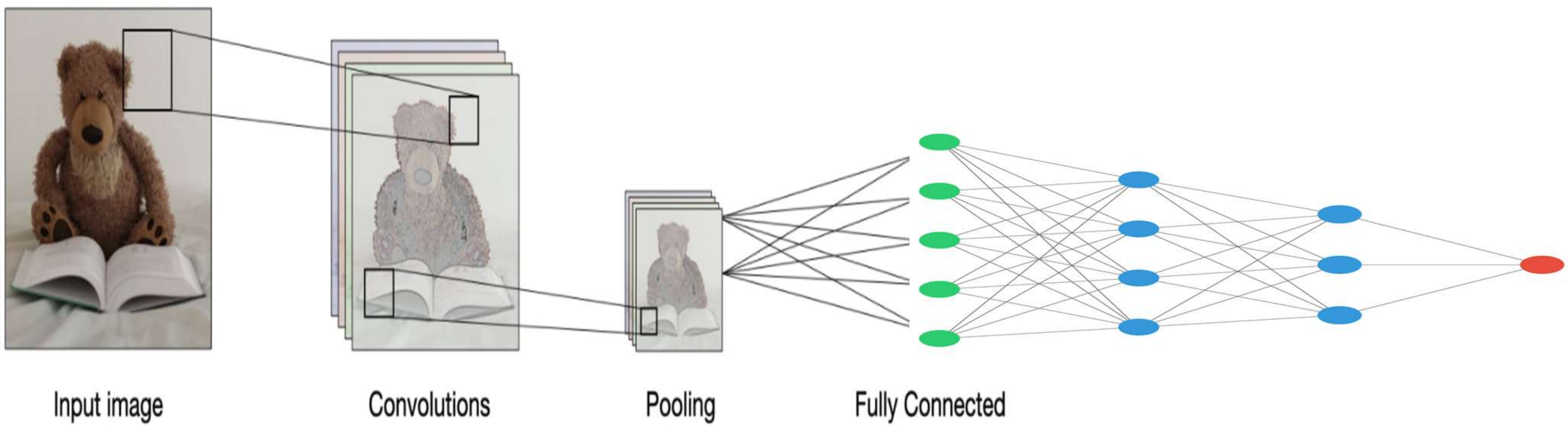
1. CONVOLUTION NEURAL NETWORK OVERVIEW

CNN ARCHITECTURE



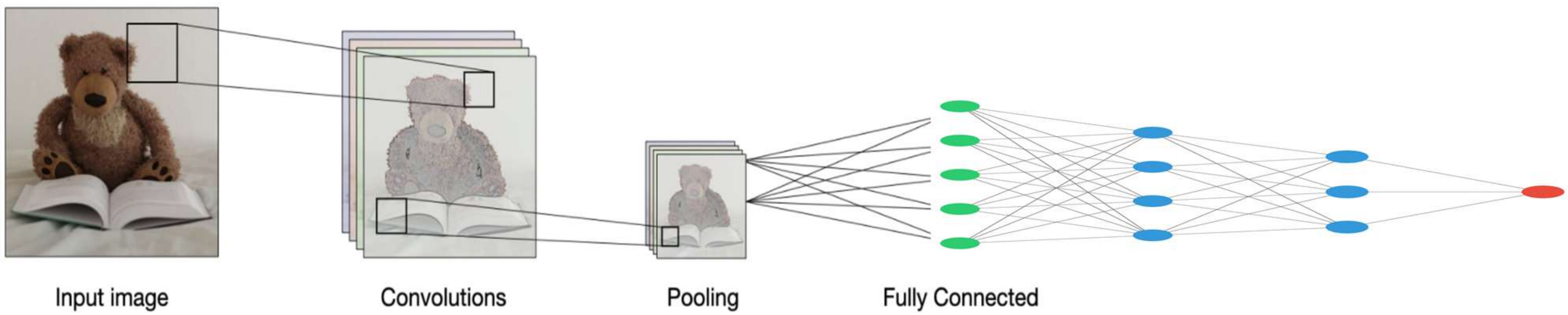
- Source: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

CNN ARCHITECTURE



- Source: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

CNN ARCHITECTURE



Input
Shape

Conv2D
SeparableConv
DepthwiseConv
ConvTranpose

MaxPooling
AveragePooling
GlobalMaxPooling
GlobalAvaragePooling

Flatten()

Dense()

Dense()

Dense()

- Source: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

3. APPLYING THE CNN WITH A CHEST IMAGE PROBLEM.

3. A CHEST IMAGE PROBLEM

a. Problem overview
b. Model 1: A simple model
c. Model 2: A model with call backs
d. Model 3: A result from a model tuning hyperparameter
e. Model 4: A model with tunned hyper-parameter without callbacks

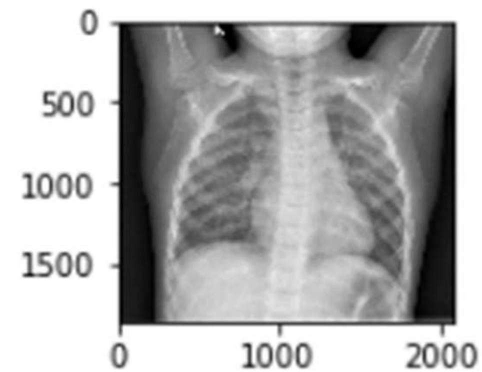
a. Problem overview

This is a supervised binary classification problem.

We want to distinguish between normal chest-mages with pneumonia infected chest images

We want to fit the model in training set, and then evaluate on the test set.

Source: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia




```
Type of the image : <class 'imageio.core.util.Array'>
Shape of the image : (1858, 2090)
Dimension of Image 2
dtype:  uint8
Maximum RGB value in this image 255
Minimum RGB value in this image 0
```

3. A CHEST IMAGE PROBLEM

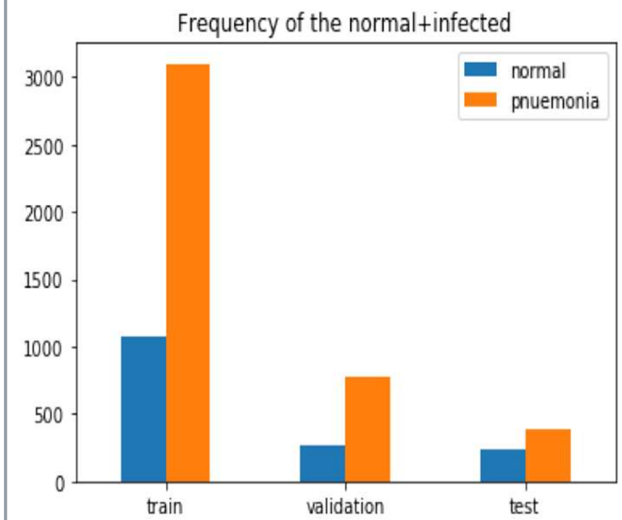
a. Problem overview
b. Model 1: A simple model
c. Model 2: A model with call backs
d. Model 3: A result from a model tuning hyperparameter
e. Model 4: A model with tunned hyper-parameter without callbacks

a. Problem overview (cont.)

The original data sets have 16 observations for validation.

In this project, I change split the train set to get new train and validation sets.

Source: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia



Frequency of the normal+infected:

	normal	pnuemonia
train	1073	3100
validation	268	775
test	234	390

3. A CHEST IMAGE PROBLEM

a. Problem overview

b. <u>Model 1: A simple model</u>

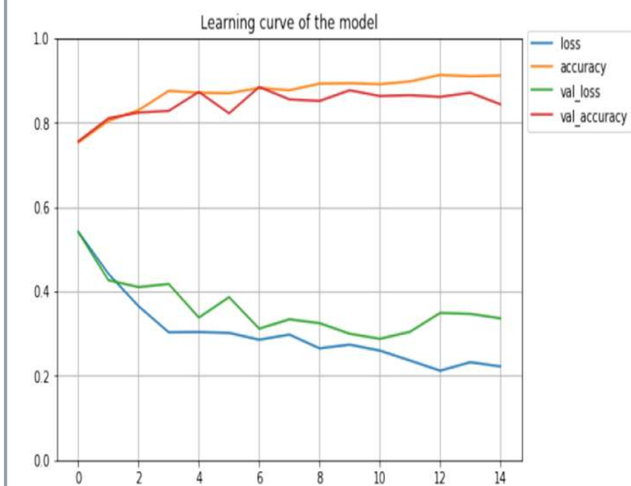
c. Model 2: A model with call backs

d. Model 3: A result from a model tuning hyperparameter

e. Model 4: A model with tunned hyper-parameter without callbacks

b. A simple model

```
model1 = keras.Sequential([
   AveragePooling2D(6,3, input_shape=(img_dims,img_dims, 3)),
   Conv2D(64, 3, activation='relu'),
   Conv2D(32, 3, activation='relu'),
   MaxPool2D(pool_size=(2,2)),  # so we use max pool to reduc
   Dropout(0.5),    # we can drop out some connection
   Flatten(),
   Dense(128, activation='relu'),  #add another dense layer b
 ↪output layer
   Dense(1, activation='sigmoid')
])
```




```
19/19 [==============================] - 9s 476ms/step - loss: 0.5184 -
accuracy: 0.7796
[0.5184427499771118, 0.7796052694320679]
Accuracy =  0.7796052694320679
```

SOME PROBLEMS TO BE CONSIDERED

- Vanishing/Exploding Gradient Problems
- Use Call back
- Overfitting
- Hyperparameter tunning

- Vanishing/Exploding Gradient Problems: the gradients/signals die out or explode and saturate.
  - Glorot and He Initialization:
  - Nonsaturating Activation Functions: *leaky ReLU*.
  - Batch Normalization

SOME PROBLEMS TO BE CONSIDERED

- Vanishing/Exploding Gradient Problems
- Use Call back
- Overfitting
- Hyperparameter tunning

- What is callback?
  - Command callbacks :
    - ModelCheckpoint: helps save checkpoint (by default) at the end of each epoch . Helps return the best model on the validation set.
    - Earlystopping: will interrupt training when it measures no progress on the validation set for a number of epochs and it will optionally roll back to the best model.
    - Why use callbacks?

      - Avoid overfitting the training set.

      - Avoid wasting time and resources):

SOME PROBLEMS TO BE CONSIDERED

- Vanishing/Exploding Gradient Problems
- Use Call back
- <u>Overfitting</u>
- Hyperparameter tunning

Overfitting

- **1. Simplifying The Model:** decrease the complexity of the model by remove layers or reduce the number of neurons.
- **2. Early Stopping:**
- **3. Use Data Augmentation:**
- **4. Use Regularization (L1, L2, Max-Norm regularization)**
- **5. Use Dropouts (with fixed dropout rate) and Monte-Carlo**

SOME PROBLEMS TO BE CONSIDERED

- Vanishing/Exploding Gradient Problems
- Use Call back
- Overfitting
- Hyperparameter tunning

Overfitting

- **1. Simplifying The Model:** decrease the complexity of the model by remove layers or reduce the number of neurons.
- **2. Early Stopping:**
- **3. Use Data Augmentation:**
- **4. Use Regularization (L1, L2, Max-Norm regularization)**
- **5. Use Dropouts (with fixed dropout rate) and Monte-Carlo**

3. A CHEST IMAGE PROBLEM

a. Problem overview
b. Model 1: A simple model
c. Model 2: A model with call backs
d. Model 3: A result from a model tuning hyperparameter
e. Model 4: A model with tunned hyper-parameter without callbacks

c. A model with call back

```
model2 = keras.Sequential([
    Conv2D(filters=16, kernel_size=(3, 3), activation='relu', padding='same', input_shape=[img_dims, img_dims, 3]),
    BatchNormalization(), # use Batch Normalization to address the vanishing/exploding gradients problems
    MaxPool2D(pool_size=(2, 2)),

    Conv2D(64, 3, activation='relu'),
    Conv2D(32, 3, activation='relu'),

    #Note: Instead of using a convolutional layer with a 5 × 5 kernel,
    # it is generally preferable to stack two layers with 3 × 3 kernels:
    # it will use less parameters and require less computations, and will usually perform better (page447 Hand_on)
    SeparableConv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same'),
    SeparableConv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    MaxPool2D(pool_size=(2, 2)),

    #Note: Note that the number of filters grows as we climb up the CNN towards the output layer (64, 128, 256) (page448 Hanh-on)
    SeparableConv2D(filters=128, kernel_size=(3, 3), activation='relu', padding='same'),
    SeparableConv2D(filters=128, kernel_size=(3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    MaxPool2D(pool_size=(2, 2)),
    Dropout(0.2),
```

```
    SeparableConv2D(filters=256, kernel_size=(3, 3), activation='relu', padding='same'),
    SeparableConv2D(filters=256, kernel_size=(3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    MaxPool2D(pool_size=(2, 2)),
    Dropout(0.2),

    #Note: we must flatten its inputs, since a dense network expects a 1D array of features for each instance
    Flatten(),
    Dense(128, activation='relu'),  #add another dense layer before doing the output layer
    Dropout(rate=0.5),     #NOte: dropout to prvent overfitting
    Dense(units=64, activation='relu'),
```

```
    Dropout(rate=0.5),
    Dense(1, activation='sigmoid')
])
```

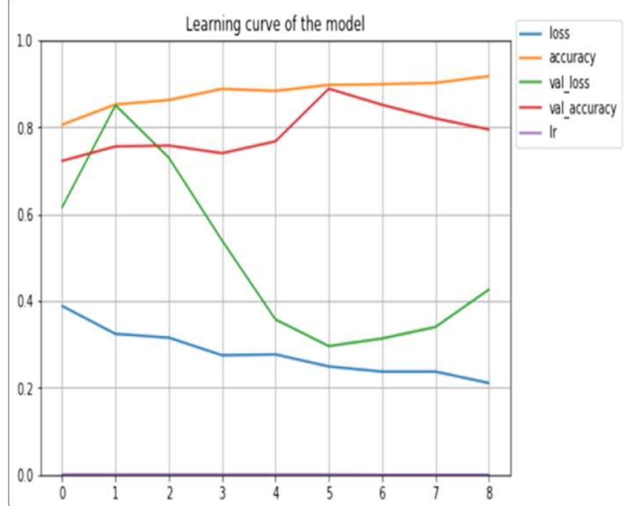
3. A CHEST IMAGE PROBLEM

a. Problem overview
b. Model 1: A simple model
c. <u>Model 2: A model with call backs</u>
d. Model 3: A result from a model tuning hyperparameter
e. Model 4: A model with tunned hyper-parameter without callbacks

c. A model with call back

```
# Callbacks
checkpoint = ModelCheckpoint(filepath='best_weights.hdf5', save_best_only=True,␣
↪save_weights_only=True)
lr_reduce = ReduceLROnPlateau(monitor='val_loss', factor=0.3, patience=4,␣
↪verbose=2, mode='max')
early_stop = EarlyStopping(monitor='val_loss', min_delta=0.1, patience=4,␣
↪mode='min')
```

```
epochs=25
model2_hist = model2.fit(
            train_generator,
            steps_per_epoch=train_generator.samples // batch_size,
            epochs=epochs,
            validation_data=valid_generator,
            validation_steps= valid_generator.samples // batch_size,
            callbacks=[checkpoint, lr_reduce, early_stop])
```




```
19/19 [==============================] - 4s 191ms/step - loss: 0.2470 -
accuracy: 0.9013
[0.24703054130077362, 0.9013158082962036]
Accuracy =  0.9013158082962036
```

SOME PROBLEMS TO BE CONSIDERED

- Vanishing/Exploding Gradient Problems
- Use Call back
- Overfitting
- Hyperparameter tunning

- Hyperparameter tunning

- number of layers,
- number of neurons per layer,
- type of activation function to use in each layer,
- weight initialization
- Learning rate, batch size

- Use GridSearchCV or RandomizedSearchCV

3. A CHEST IMAGE PROBLEM

a. Problem overview
b. Model 1: A simple model
c. Model 2: A model with call backs
d. Model 3: A result from a model tuning hyperparameter
e. Model 4: A model with tunned hyper-parameter without callbacks

d. A model with tuned hyperparameter with callbacks

```
def build_model_withcallback(hp):
    """hp: stands for hyper-parameters"""
    model = keras.Sequential()

    model.add(Conv2D(filters=16, kernel_size=(3, 3), activation='relu',
 ↪padding='same', input_shape=[img_dims, img_dims, 3]))
    model.add(BatchNormalization()) # use Batch Normalization to address
 ↪the vanishing/exploding gradients problems
    model.add(MaxPool2D(pool_size=(2, 2)))

    for i in range(hp.Int("Conv Layers", min_value=0, max_value=2)):
        model.add(keras.layers.Conv2D(hp.Choice(f"layer_{i}_filters",
 ↪[64,128,256]), 3, activation='relu', padding='same'))
    model.add(keras.layers.MaxPool2D(2,2))
    model.add(keras.layers.Dropout(0.5))

    for i in range(hp.Int("Conv Layers", min_value=0, max_value=2)):
        model.add(keras.layers.Conv2D(hp.Choice(f"layer_{i}_filters",
 ↪[64,128,256]), 3, activation='relu', padding='same'))
    model.add(keras.layers.MaxPool2D(2,2))
    model.add(keras.layers.Dropout(0.5))

    # Flatten
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(hp.Choice("Dense layer", [64, 128, 256]),
 ↪activation='relu'))
    model.add(keras.layers.Dense(1, activation='sigmoid'))
    # Compile
    model.compile(optimizer='adam',
            loss='binary_crossentropy',
            metrics=['accuracy'])
    return model
```

```
tuner_withcallback = RandomSearch(
    build_model_withcallback,
    objective='val_accuracy',
    max_trials=20,
)

tuner_withcallback.search(train_generator,
            steps_per_epoch=train_generator.samples//batch_size,
            validation_data=valid_generator,
            epochs=15,
            batch_size=batch_size,
            callbacks=[tf.keras.callbacks.EarlyStopping(patience=2)])
```

3. A CHEST IMAGE PROBLEM

a. Problem overview
b. Model 1: A simple model
c. Model 2: A model with call backs
d. <u>Model 3: A result from a model tuning hyperparameter</u>
e. Model 4: A model with tunned hyper-parameter without callbacks

d. A model with tuned hyperparameter with callbacks

```
Best val_accuracy So Far: 0.923298180103302
Total elapsed time: 05h 10m 37s
INFO:tensorflow:Oracle triggered exit
```

```
19/19 [==============================] - 8s 406ms/step - loss: 0.5907 -
accuracy: 0.8026
[0.5906840562820435, 0.8026315569877625]
```

3. A CHEST IMAGE PROBLEM

a. Problem overview
b. Model 1: A simple model
c. Model 2: A model with call backs
d. Model 3: A result from a model tuning hyperparameter
e. <u>Model 4: A model with tunned hyper-parameter without callbacks</u>

a. <u>A model with tunned hyperparameter without callbacks</u>

```
Best val_accuracy So Far: 0.9587727785110474
Total elapsed time: 12h 46m 41s
INFO:tensorflow:Oracle triggered exit
```

```
39/39 [==============================] - 16s 392ms/step - loss: 0.4852 -
accuracy: 0.8674
[0.3162696361541748, 0.9086538553237915]
```

SUMMARY THE RESULT:

1. Simple model	2. A model with call backs	3. A model with tuned hyper-parameter with callbacks	4. A model with tuned hyper-parameter without callbacks
• Valid accuracy: 0.8438 • Test accuracy: 0.7796	• Valid accuracy: 0.8887 • Test accuracy: 0.9013	• Best Valid accuracy: 0.9232 • Test accuracy: 0.8026	• Best Valid accuracy: 0.9587 • Test accuracy: 0.9086

SOME PROBLEMS THAT I NEED TO LEARN

- Not use data augmentation to increase the size of training set
- Convolution layer design
- Learning rate
- An approach getting 100% accuracy using Fast AI, a course of Jeremy Howard.

https://www.youtube.com/watch?v=gGxe2mN3kAg&list=RDCMUCX7Y2qWriXpqocG97SFW2OQ&start_radio=1&rv=gGxe2mN3kAg&t=0&ab_channel=JeremyHoward%28youtubeaccount%29JeremyHoward%28youtubeaccount%29

- Use Tensor-board to optimize CNN model

REFERENCE

- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, by Aurélien Géron, Released September 2019
- Deep Learning with Python, Second Edition, François Chollet, MEAP began March 2020
- http://cs231n.stanford.edu/
- https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks
- https://www.kaggle.com/jagadish13/keras-nn-x-ray-predict-pneumonia
- [A solution, get 100%, using Fastai] https://towardsdatascience.com/fastai-bag-of-tricks-experiments-with-a-kaggle-dataset-part-1-135e46da72f2