# Detection of Covid-19 Pneumonia via Chest X-Rays

**BCF 1**
Elangovan Karthikeyan
Lam Wei Lin, Zoey
Lee Pei Yee

https://www.youtube.com/watch?v=Gc0bl5ydCA8&ab_channel=GohYuChen

# Rubrics For Your Consideration

10% for coming up with your own **problem definition based on a dataset**
10% for **data preparation and cleaning** to **suit the problem** of your choice
20% for **exploratory data analysis**/visualization to gather **relevant insights**
20% for the use of **machine learning techniques** to solve specific problem
20% for the presentation of **data-driven insights** and the **recommendations**
10% for the quality of your final team presentation and overall impressions
10% for learning **something new and doing something beyond** this course

| Problem Definition | Data Preparation | Exploratory Data Analysis | Machine learning | Insights |
|---|---|---|---|---|

**First 2 minutes (approx.)** of your presentation -- *Your Motivation*

- You MUST mention **which dataset(s) you used**, and **what EXACTLY is your problem definition.**
- This is your chance to explain your MOTIVATION. Don't get carried away; be precise and clear.

**Next 3 minutes (approx.)** of your presentation -- *Set The Stage*

- Present your Exploratory Data Analysis and some initial data-driven Insights from the dataset.
- You MAY also mention how you are planning to set up the Analysis / ML problem for this case.
- You MUST mention how you **collected / curated / cleaned / prepared the data for this problem**.
- Did you only use tools and techniques learned in this course? What ELSE did you learn / try?

**Next 3 minutes (approx.)** of your presentation -- *Core Analysis*

- If you used ML (regression, classification, or something else); mention mainly WHICH one(s).
- You may now briefly CLARIFY **why and how the ML problem(s) aim(s) to solve your objective**.
- How did you apply ML technique(s) to SOLVE your problem? Which **model(s), how and why**?
- Did you only use tools and techniques learned in this course? What ELSE did you learn / try?

**Last 2 minutes (approx.)** of your presentation -- *Finish Strong*

- What is the **OUTCOME** of your project? Did it **solve your original problem**? Anything interesting?
- What are your data-driven INSIGHTS and recommendations / views towards the target problem?

*Everything you plan to showcase should be presented within the 10 mins. Plan carefully and smartly.*
Present the main aspects of your project and data-story in the 10 mins; extra items may be on GitHub.
No need to mention who in your team worked on which part of the project -- this should be on GitHub.
No need to cite references to other related works in your presentation -- this should be on GitHub too.

Your entire codebase for the project should be posted on GitHub. You will submit the link to us.
Detailed instructions on preparing the GitHub repository will be posted here, closer to deadline.
Note that you DO NOT need to collaborate on GitHub. Posting your final codes will be enough.

# Current Situation

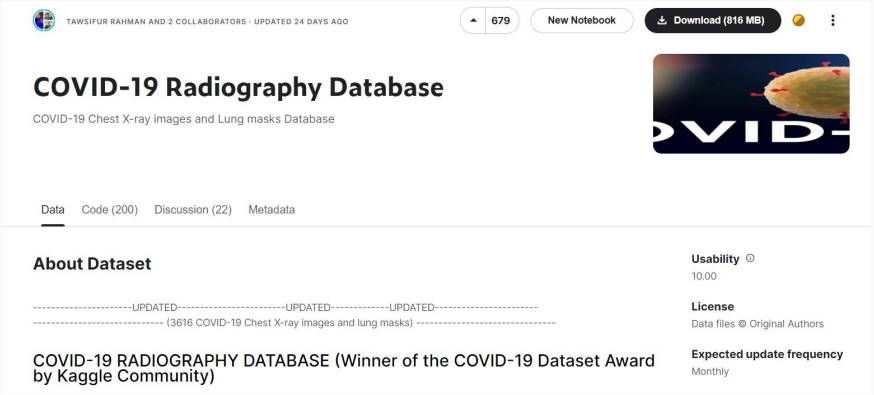
As COVID-19 becomes endemic, the **strain** on healthcare services worldwide **intensifies**, making it more difficult to **manage & detect** the serious complications of COVID-19.

We see the possibility of replacing the current, manual way of diagnosing pneumonia which is **unnecessarily exhausting limited resources** and **time-consuming** yet may not be highly effective due to the presence of **inevitable human errors**

# How can we improve the efficiency of diagnosing normal, viral and covid-induced pneumonia in healthcare settings?

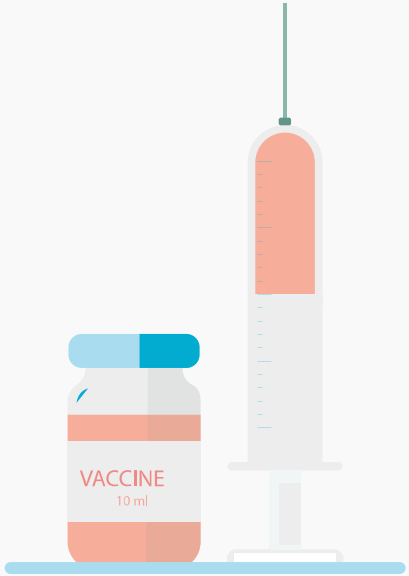
# PROBLEM DEFINITION

## COVID-19 Chest X-Ray Image Dataset




- Found a dataset containing COVID-19 Chest X-Rays on Kaggle
- 21165 Chest X-Rays from 4 different categories
  - COVID_19
  - Normal
  - Lung Opacity
  - Viral Pneumonia

# Our Solution...




To tackle this Multi-Class Classification Problem, we propose:

**Image Classification model to classify Chest X-Ray images into 3 classes (normal, viral pneumonia and COVID-induced pneumonia)**

**Approach:**

- Clean and use only relevant data
- Split into train, validation and test set
- Apply Transfer Learning

# DATA PREPARATION

```
# Basic Libraries
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt # we only need pyplot
sb.set() # set the default Seaborn style for graphics
```

```
[2] #Testing for GPU
    import tensorflow as tf
    device_name = tf.test.gpu_device_name()
    if device_name != '/device:GPU:0':
      raise SystemError('GPU device not found')
    print('Found GPU at: {}'.format(device_name))
```

```
Found GPU at: /device:GPU:0
```

**Import Essential Libraries for analysis and machine learning such as Tensorflow (NEW!)**

Importing Dataset using Kaggle API (kaggle.json file required)

```
[3] !pip install -q kaggle
    from google.colab import files
    files.upload()
    ! mkdir ~/.kaggle
    ! cp kaggle.json ~/.kaggle/
    ! chmod 600 ~/.kaggle/kaggle.json
    ! kaggle datasets list
```

**Imported Dataset using Kaggle API**

# DATA PREPARATION...

```
[16] files_normal = glob.glob('./COVID-19_Radiography_Dataset/Normal/images/*.png')
     files_COVID = glob.glob('./COVID-19_Radiography_Dataset/COVID/images/*.png')
     files_viral = glob.glob('./COVID-19_Radiography_Dataset/Viral Pneumonia/images/*.png')
     files_Lung = glob.glob('./COVID-19_Radiography_Dataset/Lung_Opacity/images/*.png')
```

```
[17] print(len(files_normal))
     print(len(files_viral))
     print(len(files_COVID))
     print(len(files_Lung))
```

```
10192
1345
3616
6012
```

## Lung Opacity

- 6012 images in dataset
- Contains a mix of Viral Pneumonia and other lung infections

Overlapping data could cause accuracy of classifier to decrease

Other lung infections are not relevant to our goal

**Solution: Filter out the data**

# DATA PREPARATION...

Cleaning Data by Checking for Full White/Black Images in Files To Be Eliminated from dataset

```
[87] from PIL import Image

     def checkFullWhiteOrBlack(folder):
       count = 0
       for i in range(len(folder)):
         img = Image.open(folder[i])
         clrs = img.getcolors()
         if(len(clrs) == 1):
           count += 1
       return count

     print("No Of Full White/Black Images in Files - Normal" , checkFullWhiteOrBlack(files_normal))
     print("No Of Full White/Black Images in Files - Viral" , checkFullWhiteOrBlack(files_viral))
     print("No Of Full White/Black Images in Files - COVID" , checkFullWhiteOrBlack(files_COVID))
```

```
No Of Full White/Black Images in Files - Normal 0
No Of Full White/Black Images in Files - Viral 0
No Of Full White/Black Images in Files - COVID 0
```

It can be observed that there are no full white/black images in all the files, hence no further cleaning is required.

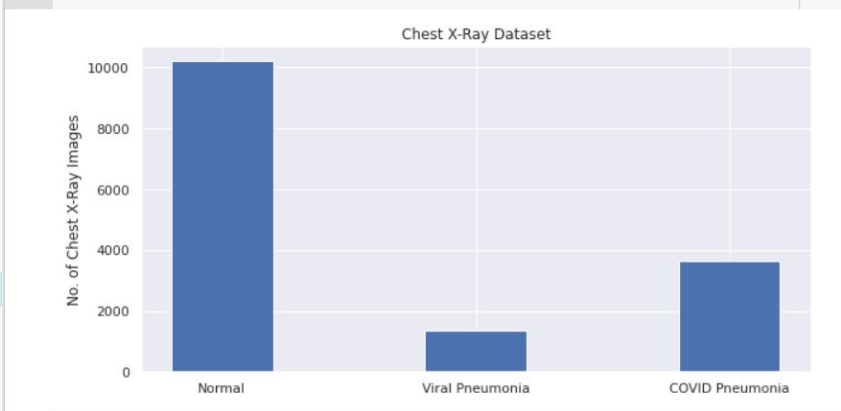
## Checking for Blank Images

- There may be full black or white images within the files
- This would interfere with the accuracy of the classifier and the parameters used in it

## Solution: Check and filter out blank images

- But upon checking, we found no blank images
- Did not filter out any image

# DATA PREPARATION . . .




```
import numpy as np
np.random.seed(42)
files_normal = np.random.choice(files_normal,1345,replace=False)
files_viral = np.random.choice(files_viral,1345,replace=False)
files_COVID = np.random.choice(files_COVID,1345,replace=False)
```

```
[18] print(len(files_normal))
     print(len(files_viral))
     print(len(files_COVID))

     1345
     1345
     1345
```

## Data Imbalance

- 10192 Normal X-Rays
- 1345 Viral Pneumonia X-Rays
- 3616 COVID Pneumonia X-Rays

Highly Skewed to Normal Dataset

## Solution: Undersampling (NEW!)

- Use class with lowest number of datapoints as base
- Randomly pick data from other classes

Balances out the number of chest x-ray images for each category

# Train Test Split

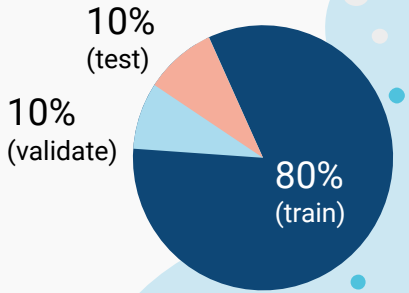
```
[ ]  from sklearn.model_selection import train_test_split

     normal_train, normal_test = train_test_split(files_normal, test_size=0.2, random_state=42, shuffle = True)
     normal_valid, normal_test = train_test_split(normal_test, test_size=0.5, random_state=42, shuffle = True)
     viral_train, viral_test = train_test_split(files_viral, test_size=0.2, random_state=42, shuffle = True)
     viral_valid, viral_test = train_test_split(viral_test, test_size=0.5, random_state=42, shuffle = True)
     COVID_train, COVID_test = train_test_split(files_COVID, test_size=0.2, random_state=42, shuffle = True)
     COVID_valid, COVID_test = train_test_split(COVID_test, test_size=0.5, random_state=42, shuffle = True)
```

```
normal-train:  1076
normal-valid:  134
normal-test:  135
Total Normal:  1345
```

```
viral-train:  1076
viral-valid:  134
viral-test:  135
Total Viral:  1345
```

```
COVID-train:  1076
COVID-valid:  134
COVID-test:  135
Total COVID:  1345
```

```
Total Train dataset:  3228
Total Valid dataset:  402
Total Test dataset:  405
```



# Encoding Classes in Data ...

```
[23] #Assigned class for normal - 0, viral - 1 , COVID - 2 - check if we wanna assign 0 to viral as well!

     df_train_normal['class'] = pd.Series([0 for x in range(len(df_train_normal.index))], index=df_train_normal.index)
     df_train_viral['class'] = pd.Series([1 for x in range(len(df_train_viral.index))], index=df_train_viral.index)
     df_train_COVID['class'] = pd.Series([2 for x in range(len(df_train_COVID.index))], index=df_train_COVID.index)

     df_test_normal['class'] = pd.Series([0 for x in range(len(df_test_normal.index))], index=df_test_normal.index)
     df_test_viral['class'] = pd.Series([1 for x in range(len(df_test_viral.index))], index=df_test_viral.index)
     df_test_COVID['class'] = pd.Series([2 for x in range(len(df_test_COVID.index))], index=df_test_COVID.index)

     df_valid_normal['class'] = pd.Series([0 for x in range(len(df_valid_normal.index))], index=df_valid_normal.index)
     df_valid_viral['class'] = pd.Series([1 for x in range(len(df_valid_viral.index))], index=df_valid_viral.index)
     df_valid_COVID['class'] = pd.Series([2 for x in range(len(df_valid_COVID.index))], index=df_valid_COVID.index)
     df_train_normal.head()
```

| | filename | class |
|---|---|---|
| 0 | ./COVID-19_Radiography_Dataset/Normal/images/N... | 0 |
| 1 | ./COVID-19_Radiography_Dataset/Normal/images/N... | 0 |
| 2 | ./COVID-19_Radiography_Dataset/Normal/images/N... | 0 |
| 3 | ./COVID-19_Radiography_Dataset/Normal/images/N... | 0 |
| 4 | ./COVID-19_Radiography_Dataset/Normal/images/N... | 0 |

**0** - Normal

**1** - Viral Pneumonia

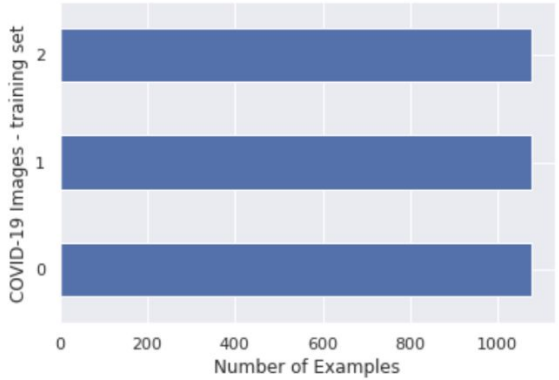
**2** - COVID Pneumonia

# Exploratory Data Analysis ...

## Inspecting the distribution of classes in the train dataset

```
print(df['class'].value_counts().sort_index())

ax = df['class'].value_counts().sort_index().plot.barh()
ax.set_xlabel("Number of Examples", fontsize=12)
ax.set_ylabel("COVID-19 Images - training set", fontsize=12)
plt.show()
```

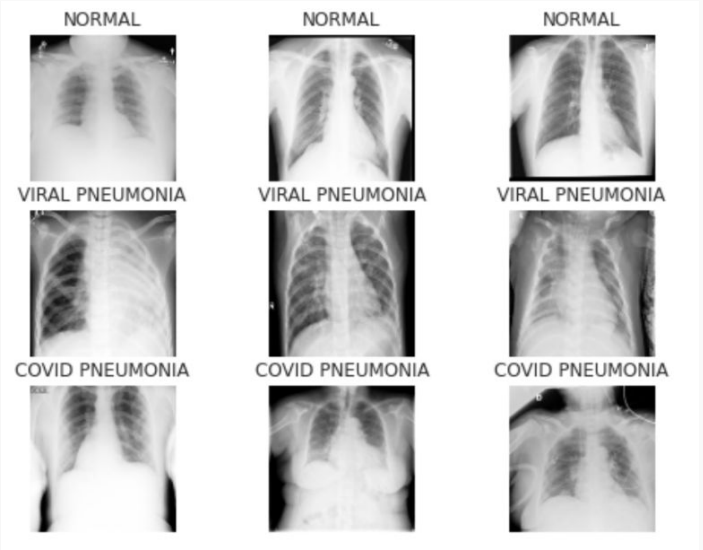
```
0    1076
1    1076
2    1076
Name: class, dtype: int64
```



# Exploratory Data Analysis

## Raw Observations Of Image Data

```
# randomly select 3 of each
select_norm = np.random.choice(train_normal, 3, replace = False)
select_viral = np.random.choice(train_viral, 3, replace = False)
select_covid = np.random.choice(train_covid, 3, replace = False)

# plotting 3 x 3 image matrix
fig = plt.figure(figsize = (8,6))
for i in range(9):
    if i < 3:
        fp = select_norm[i]
        label = 'NORMAL'
    elif i < 6:
        fp = select_viral[i -3]
        label = 'VIRAL PNEUMONIA'
    else:
        fp = select_covid[i - 6]
        label = 'COVID PNEUMONIA'
    ax = fig.add_subplot(3, 3, i+1)

    # to plot without rescaling, remove target_size
    fn = image.load_img(fp, target_size = (100,100), color_mode='grayscale')
    plt.imshow(fn, cmap='Greys_r')
    plt.title(label)
    plt.axis('off')
plt.show()
```



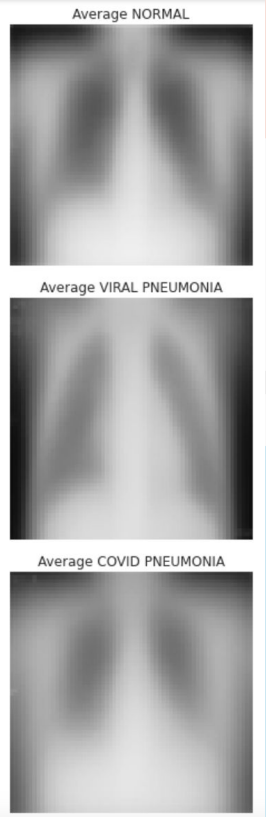
# Exploratory Data Analysis ...

## Average Comparison Of Image Data(NEW!)

Converting Images to Matrix → Finding the average image for each class

```
def img2np(list_of_filename, size = (64, 64)):
    # iterating through each file
    for fn in list_of_filename:
        fp = fn
        current_image = image.load_img(fp, target_size = size,
                                       color_mode = 'grayscale
        # covert image to a matrix
        img_ts = image.img_to_array(current_image)
        # turn that into a vector / 1D array
        img_ts = [img_ts.ravel()]
        try:
            # concatenate different images
            full_mat = np.concatenate((full_mat, img_ts))
        except UnboundLocalError:
            # if not assigned yet, assign one
            full_mat = img_ts
    return full_mat

# run it on our folders
normal_images = img2np(train_normal)
viral_images = img2np(train_viral)
covid_images = img2np(train_covid)
```

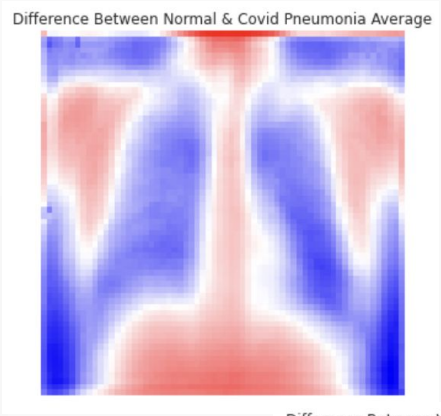
```
def find_mean_img(full_mat, title, size = (64, 64)):
    # calculate the average
    mean_img = np.mean(full_mat, axis = 0)
    # reshape it back to a matrix
    mean_img = mean_img.reshape(size)
    plt.imshow(mean_img, vmin=0, vmax=255, cmap='Greys_r')
    plt.title(f'Average {title}')
    plt.axis('off')
    plt.show()
    return mean_img

norm_mean = find_mean_img(normal_images, 'NORMAL')
viral_mean = find_mean_img(viral_images, 'VIRAL PNEUMONIA')
covid_mean = find_mean_img(covid_images, 'COVID PNEUMONIA')
```

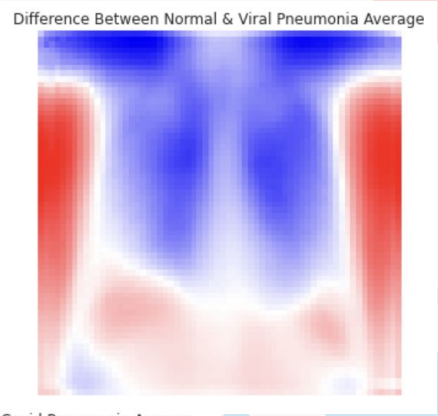


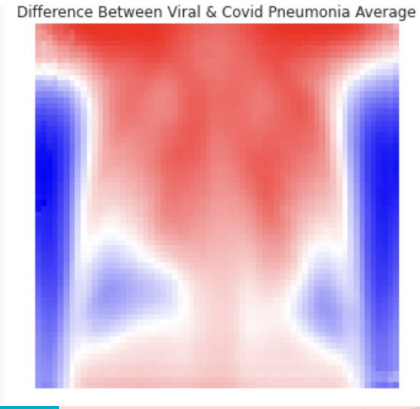
# Exploratory Data Analysis

## Contrast between Average Images Of Normal, Viral Pneumonia and Covid Pneumonia(NEW!)

```
[38] fig = plt.figure(figsize = (18,6))
     for i in range(3):
         if i == 0:
             contrast_mean = norm_mean - viral_mean
             label = 'Difference Between Normal & Viral Pneumonia Average'
         elif i == 1:
             contrast_mean = norm_mean - covid_mean
             label = 'Difference Between Normal & Covid Pneumonia Average'
         else:
             contrast_mean = viral_mean - covid_mean
             label = 'Difference Between Viral & Covid Pneumonia Average'

         ax = fig.add_subplot(1, 3, i+1)

         plt.imshow(contrast_mean, cmap='bwr',)
         plt.title(label)
         plt.axis('off')
     plt.show()
```

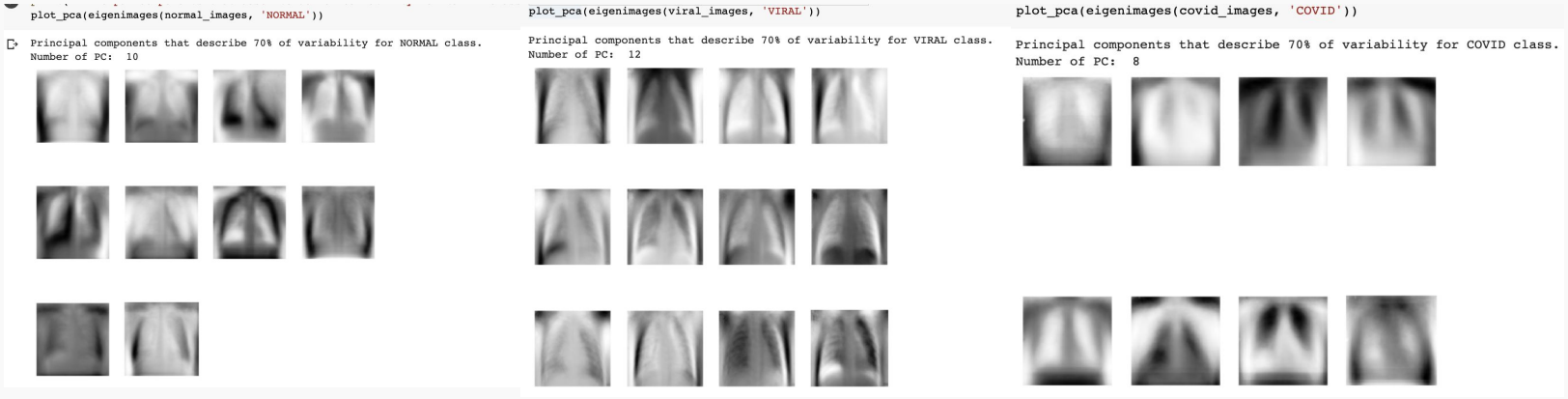


# Exploratory Data Analysis ...

## Eigenimages from Principal Component Analysis (PCA) (NEW!)

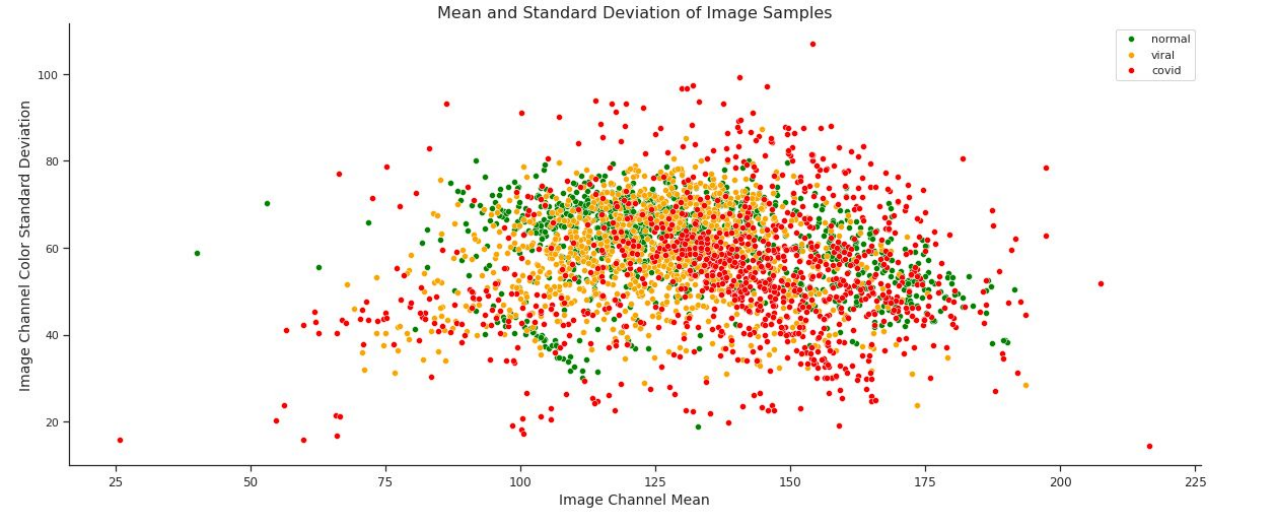



### Dimension reduction technique — Principal Component Analysis (PCA)

- Visualize the components that best describes each class.
- Eigenimages — Plotted from matrix shaped by components from PCA
  - Principal components that describe **70%** of variability for each class.

# Exploratory Data Analysis ...

## Image Value Distribution (NEW!)




3 clusters — Normal, Viral, Covid

**Normal:**

- Mean - 100 to 150
- Standard Deviation - 60 to 70

**Viral:**

- Mean - 100 to 150
- Standard Deviation - 50 to 80

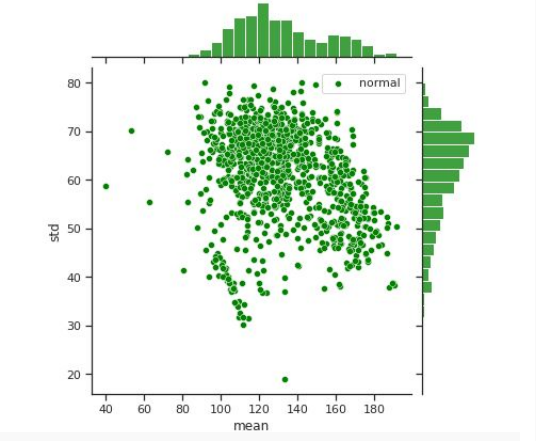
**Covid:**

- Mean - 125 to 175
- Standard Deviation - 40 to 60

# Exploratory Data Analysis

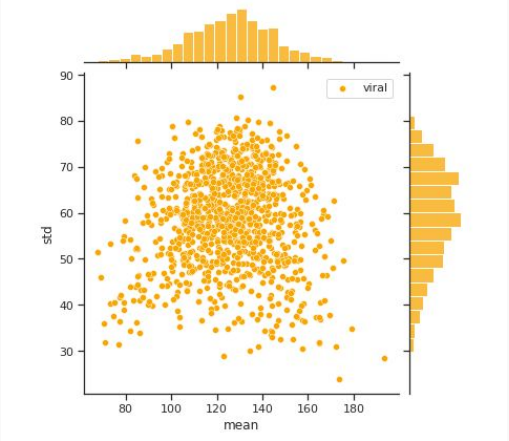
## Image Value Distribution (NEW!)

**black - rgb(0,0,0)**
**White - rgb(255,255,255)**



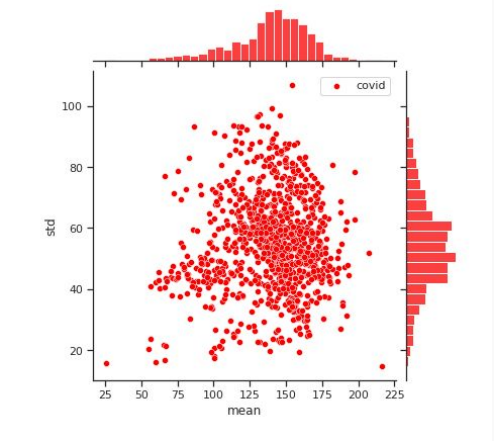

**Normal:**

- 100 to 150
- Majority: 120 to 125




**Viral:**

- 105 to 145
- Majority: 130 to 135



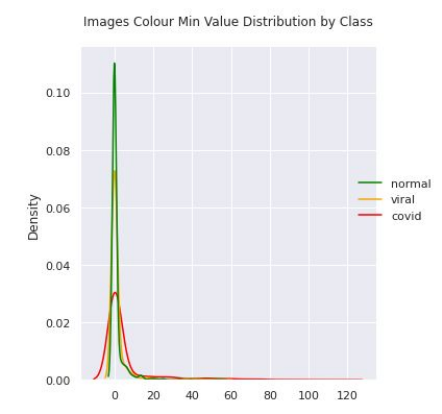

**Covid:**

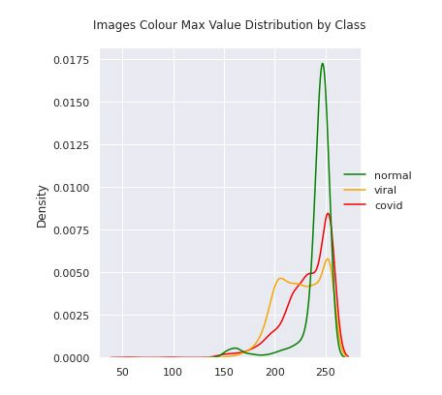
- 125 to 175
- Majority: 145 to 150

# Exploratory Data Analysis

## Image Value Distribution — Pixel Density plots (NEW!)

**black - rgb(0,0,0)**
**White - rgb(255,255,255)**


### Mean Values:

- Normal: 120
- Viral: 130
- Covid: 145

### Min Values:

- Normal: 0
- Viral: 0
- Covid: 0

### Max Values:

- Normal: 245
- Viral: 200 to 250
- Covid: 230 to 250

# Machine Learning...

## Preparation of Pipeline to Load images

```
# convert the dataframe into 2 lists to use for filename and labels
train_filenames_list = df["filename"].tolist()
train_labels_list = df["class"].astype('int32').tolist()

# convert the dataframe into 2 lists to use for filename and labels
val_filenames_list = df_val["filename"].tolist()
val_labels_list = df_val["class"].astype('int32').tolist()

# convert the dataframe into 2 lists to use for filename and labels
test_filenames_list = df_test["filename"].tolist()
test_labels_list = df_test["class"].astype('int32').tolist()

#number of classes
num_classes = 3
```

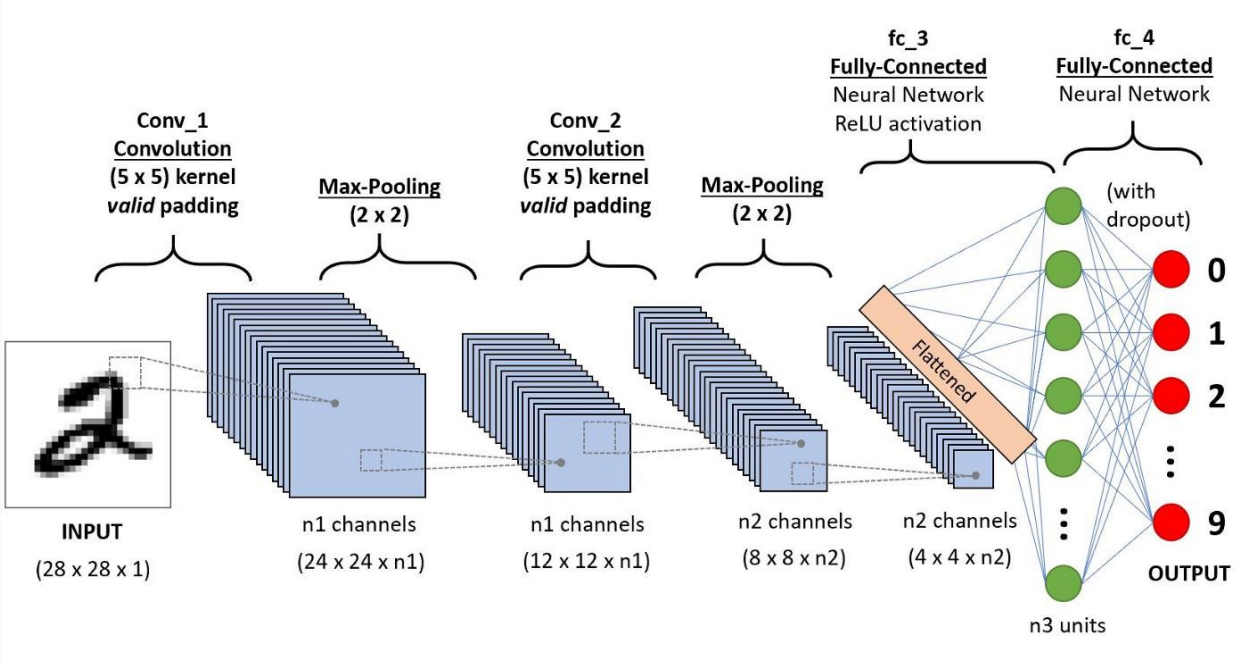
```
# Reads an image from a file, decodes it into a tensor, and resizes it
# to a fixed shape.
img_rows, img_cols = 299,299

def _parse_function(filename, label):
  image_string = tf.io.read_file(filename)
  image_decoded = tf.image.decode_jpeg(image_string,channels=3)
  image_resized = tf.image.resize(image_decoded, [img_rows, img_cols])
  label = tf.one_hot(label, num_classes)
  return image_resized, label
```

```
[46] train_dataset = tf.data.Dataset.from_tensor_slices((filenames, labels))
     train_dataset = train_dataset.map(_parse_function)
     train_dataset = train_dataset.repeat(100)
     train_dataset = train_dataset.batch(32)
```

```
[47] valid_dataset = tf.data.Dataset.from_tensor_slices((val_filenames, val_labels))
     valid_dataset = valid_dataset.map(_parse_function)
     valid_dataset = valid_dataset.repeat(100)
     valid_dataset = valid_dataset.batch(32)
```

```
[48] test_dataset = tf.data.Dataset.from_tensor_slices((test_filenames, test_labels))
     test_dataset = test_dataset.map(_parse_function)
     test_dataset = test_dataset.repeat(100)
     test_dataset = test_dataset.batch(32)
```

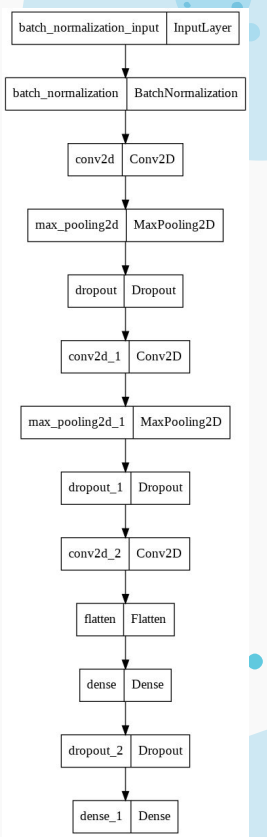
# Convolutional Neural Networks (CNN) (NEW!)



# Convolutional Neural Networks (CNN) (NEW!)

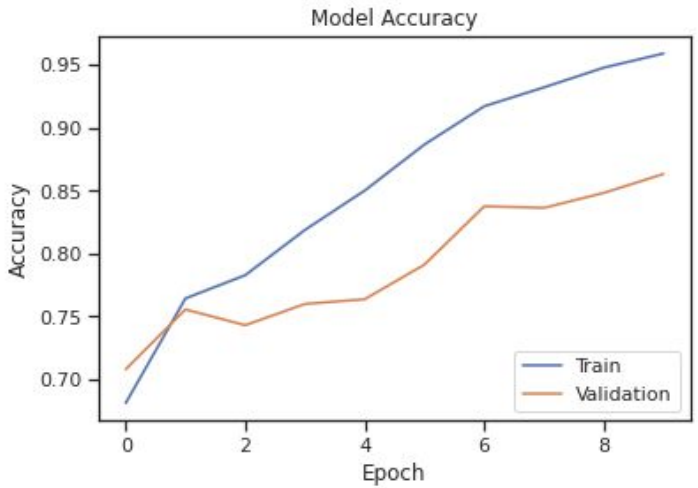
```
[55] from tensorflow.keras import layers, models

    cnn_model = tf.keras.models.Sequential()
    cnn_model.add(layers.BatchNormalization(input_shape=(299,299,3)))
    cnn_model.add(layers.Conv2D(filters = 128, kernel_size = (3, 3), activation = 'relu'))
    cnn_model.add(layers.MaxPooling2D((2, 2)))
    cnn_model.add(layers.Dropout(0.3))

    cnn_model.add(layers.Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu'))
    cnn_model.add(layers.MaxPooling2D((2, 2)))
    cnn_model.add(layers.Dropout(0.35))

    cnn_model.add(layers.Conv2D(filters = 32, kernel_size = (3, 3), activation = 'relu'))
    cnn_model.add(layers.Flatten())
    cnn_model.add(layers.Dense(units = 128, activation = 'relu'))
    cnn_model.add(layers.Dropout(0.3))

    cnn_model.add(layers.Dense(units = 3, activation = 'softmax'))

    cnn_model.compile(optimizer = 'adam',
              loss = 'categorical_crossentropy',
               metrics = ['accuracy'])

    cnn_model.summary()
```

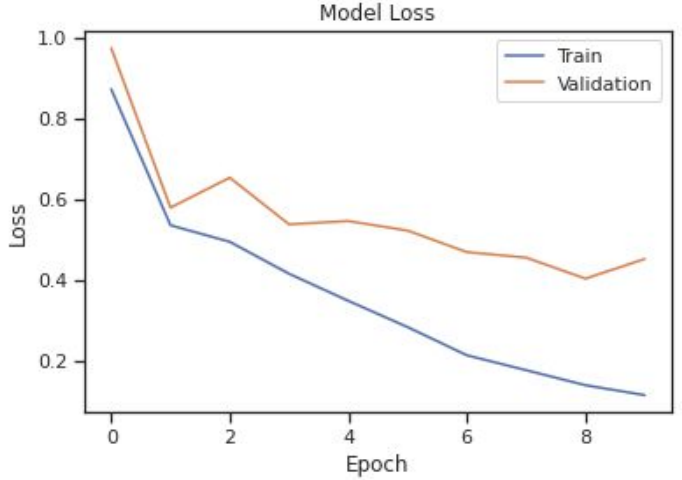
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 batch_normalization (BatchN  (None, 299, 299, 3)      12
 ormalization)

 conv2d (Conv2D)             (None, 297, 297, 128)     3584

 max_pooling2d (MaxPooling2D  (None, 148, 148, 128)    0
 )

 dropout (Dropout)           (None, 148, 148, 128)     0

 conv2d_1 (Conv2D)           (None, 146, 146, 64)      73792

 max_pooling2d_1 (MaxPooling  (None, 73, 73, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 73, 73, 64)        0

 conv2d_2 (Conv2D)           (None, 71, 71, 32)        18464

 flatten (Flatten)           (None, 161312)            0

 dense (Dense)               (None, 128)               20648064

 dropout_2 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 3)                 387

=================================================================
Total params: 20,744,303
Trainable params: 20,744,297
Non-trainable params: 6
_________________________________________________________________
```



# Convolutional Neural Networks (CNN) (NEW!)

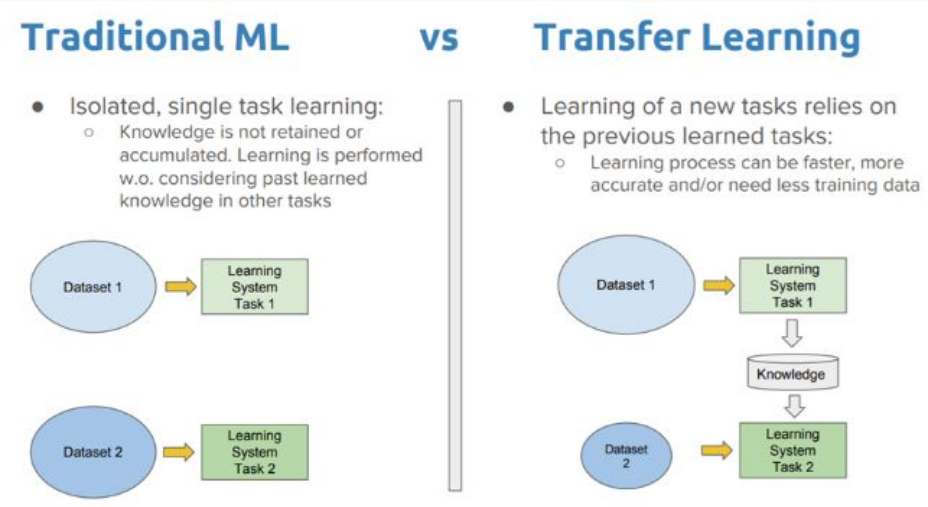
```
[59] plot_loss_acc(history_cnn)
```




```
[60] metrics = cnn_model.evaluate(test_dataset,steps=50)
     print("model accuracy:",metrics[1])

     50/50 [==============================] - 7s 144ms/step - loss: 0.3920 - accuracy: 0.8475
     model accuracy: 0.8475000262260437
```

# Transfer Learning(NEW!)

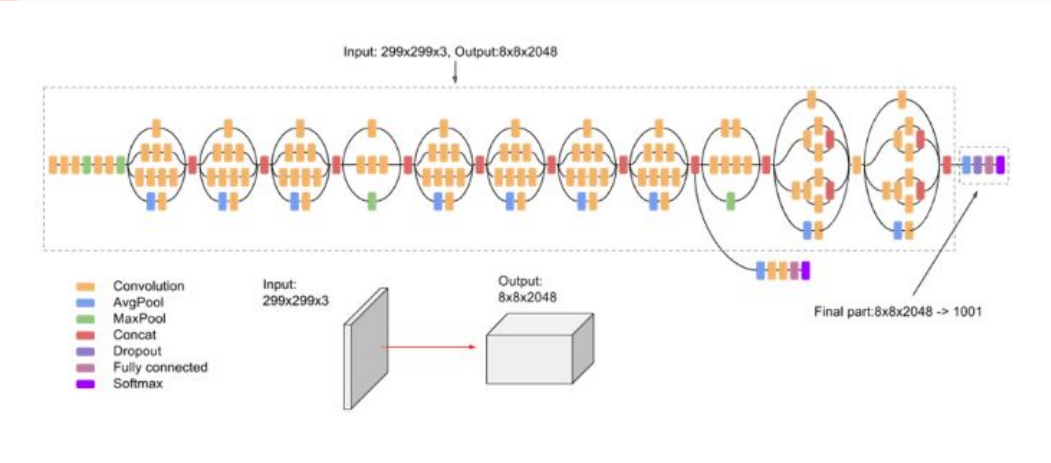



## Key Strengths Over Traditional ML models

- Don't require training from **scratch**
- **Computationally less expensive** and efficient to achieve better results
- Achieve **optimal performance faster** as they leverage knowledge from previously trained models that already understand the features

# Transfer Learning(NEW!)

## Pre-trained model Used : Inception V3




**Key aspects:**

- Good accuracy
- Low parameters

Feasible to deploy this **lightweight** model in **healthcare** settings

# Transfer Learning ...

1. Creating a base model from InceptionV3

```
[61] # InceptionV3 Model - trained on image net
     base_model = InceptionV3(include_top=False, weights='imagenet')

     Downloading data from https://storage.googleapis.com/tensorflow/k
     87916544/87910968 [==============================] - 1s 0us/step
     87924736/87910968 [==============================] - 1s 0us/step
```

```
=================================================
Total params: 21,802,784
Trainable params: 21,768,352
Non-trainable params: 34,432
_________________________________________________
```

2. Creating a simple classification head so that we can train it

```
#add a global spatial average pooling layer
x = base_model.output

x = GlobalAveragePooling2D()(x)

# let's add a fully-connected layer
x = Dense(1024, activation='relu')(x)

# 3 Classes for Detection of Pneumonia
num_classes = 3
predictions = Dense(num_classes, activation='softmax')(x)
```

# Transfer Learning ...

3. Freezing convolutional base layers except BatchNorm, to use as a feature extractor

```
# Creating the model that we will train
model = Model(inputs=base_model.input, outputs=predictions)
```

```
# Freeze the InceptionV3 layers except the BatchNorm
for layer in base_model.layers:
    if 'batch' in layer.name:
      print(layer.name)
      layer.trainable = True
    else:
      layer.trainable = False
```

4. Selecting an Optimizer - **The Adam Optimization algorithm**

```
# tf.keras optimizer
opt = tf.keras.optimizers.Adam(lr)
```

5. Compiling The Model with Loss Function

```
# compile the model
model.compile(optimizer= opt, loss='categorical_crossentropy',metrics=['accuracy'])
```

# Transfer Learning ...

## 6. Creating Checkpoints To Save Best Weights

```
#Creating Checkpoints to save best weights
checkpoint2 = ModelCheckpoint('./checkpoints/best_weights_InceptionV3Model.hdf5', verbose=1, save_best_only=True, mode='auto')
```

## 7. Setting up some hyperparameters

```
# Set up some Hyperparameters

batch_size = 32
lr = 0.001
```

```
training_data_size = len(normal_train) + len(viral_train)+len(COVID_train)

train_steps = int(training_data_size/batch_size) #training data / batch size = 3228/32
print(train_steps)
val_steps = 50
epochs = 5
```

```
100
```

# Transfer Learning ...

## Fitting the model

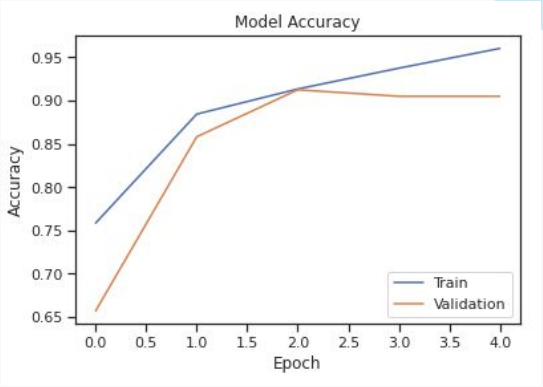
```
history = model.fit( train_dataset, steps_per_epoch = train_steps,
                    epochs = epochs,
                    validation_data = valid_dataset,
                    validation_steps = val_steps,
                    callbacks=[checkpoint2])
```

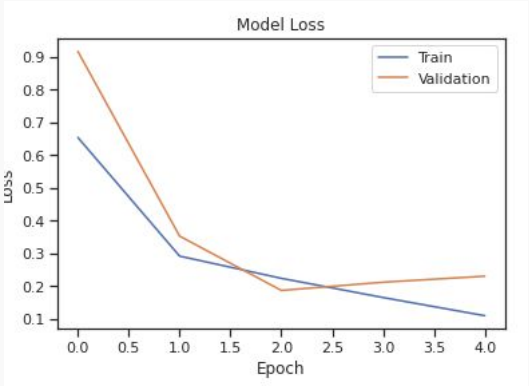
```
Epoch 1/5
100/100 [==============================] - ETA: 0s - loss: 0.6543 - accuracy: 0.7584
Epoch 1: val_loss improved from inf to 0.91719, saving model to ./checkpoints/best_weights_InceptionV3Model.hdf5
100/100 [==============================] - 55s 463ms/step - loss: 0.6543 - accuracy: 0.7584 - val_loss: 0.9172 - val_accuracy: 0.6569
Epoch 2/5
100/100 [==============================] - ETA: 0s - loss: 0.2919 - accuracy: 0.8844
Epoch 2: val_loss improved from 0.91719 to 0.35303, saving model to ./checkpoints/best_weights_InceptionV3Model.hdf5
100/100 [==============================] - 45s 447ms/step - loss: 0.2919 - accuracy: 0.8844 - val_loss: 0.3530 - val_accuracy: 0.8581
Epoch 3/5
100/100 [==============================] - ETA: 0s - loss: 0.2241 - accuracy: 0.9134
Epoch 3: val_loss improved from 0.35303 to 0.18712, saving model to ./checkpoints/best_weights_InceptionV3Model.hdf5
100/100 [==============================] - 45s 447ms/step - loss: 0.2241 - accuracy: 0.9134 - val_loss: 0.1871 - val_accuracy: 0.9125
Epoch 4/5
100/100 [==============================] - ETA: 0s - loss: 0.1654 - accuracy: 0.9375
Epoch 4: val_loss did not improve from 0.18712
100/100 [==============================] - 44s 439ms/step - loss: 0.1654 - accuracy: 0.9375 - val_loss: 0.2122 - val_accuracy: 0.9050
Epoch 5/5
100/100 [==============================] - ETA: 0s - loss: 0.1099 - accuracy: 0.9603
Epoch 5: val_loss did not improve from 0.18712
100/100 [==============================] - 44s 438ms/step - loss: 0.1099 - accuracy: 0.9603 - val_loss: 0.2306 - val_accuracy: 0.9050
```

# Transfer Learning ...

## Visualising Loss Over Epochs

```
#Visualising model loss over epochs

def plot_train(hist):
    h = hist.history
    if 'acc' in h:
        meas='acc'
        loc='lower right'
    else:
        meas='loss'
        loc='upper right'
    plt.plot(hist.history[meas])
    plt.plot(hist.history['val_'+meas])
    plt.title('model '+meas)
    plt.ylabel(meas)
    plt.xlabel('epoch')
    plt.legend(['train', 'validation'], loc=loc)
```



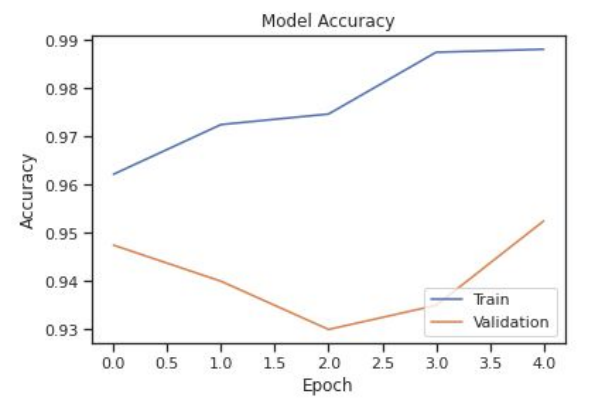

**Increase in Validation Accuracy**
**Drop in Validation Loss**

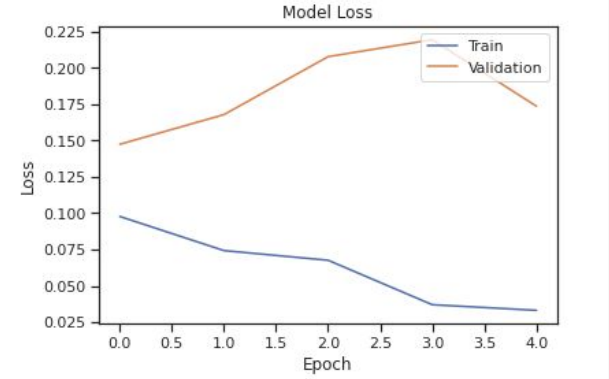
# Transfer Learning ...

Further training for more epochs using more callbacks for optimisation in training process

```
reduce_LR = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',factor = 0.9, patience=2,cooldown=1, min_lr = 0.00001)
lr_print = tf.keras.callbacks.LambdaCallback(on_epoch_begin=lambda epoch,logs: print("lr:",K.eval(model.optimizer.lr)))
early_stopping = tf.keras.callbacks.EarlyStopping(monitor = 'val_loss', mode = 'min', verbose = 1, patience = 4)
```

```
# # Train the model with validation
history2 = model.fit ( train_dataset, steps_per_epoch = train_steps,
                    epochs = 10,
                    validation_data = valid_dataset,
                    validation_steps = val_steps,
                    callbacks=[checkpoint2,lr_print,reduce_LR,early_stopping])
```




**Some improvement in validation accuracy**

- **Some signs of overfitting**

# Outcomes ...

## Evaluating model with best weights:

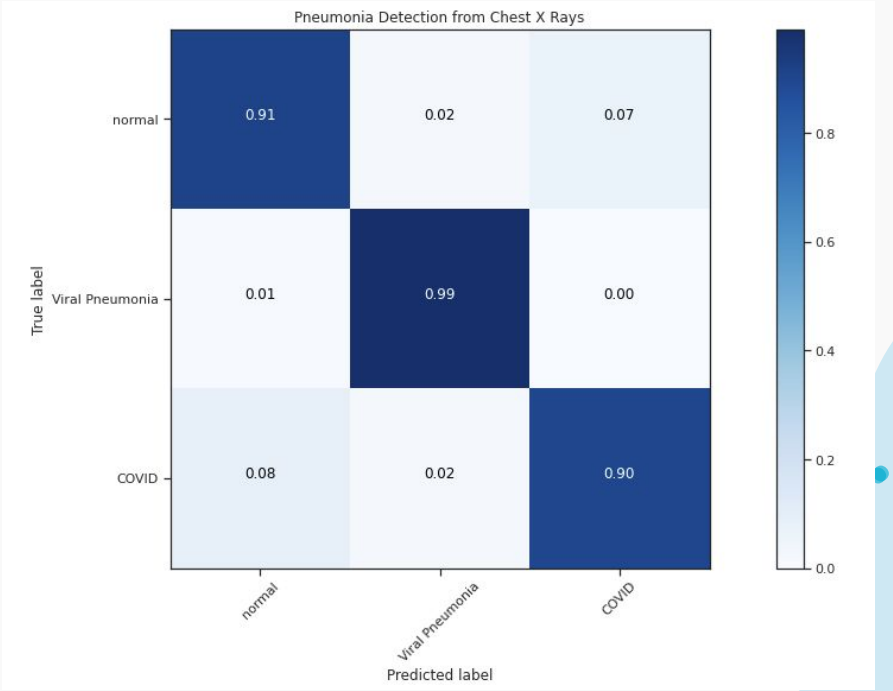
- Test Accuracy: 0.9350000023841858
- Outstanding accuracy of 93.5%.

## Confusion Matrix:

- Our model has a **significantly high true positives** seen from 0.91(normal), 0.99(viral) and 0.90(COVID)
- There is a relatively significant chance of classifying **viral pneumonia** (0.01) and **COVID X-Rays** (0.08) as **normal** (0.09)
- The chances of wrongly classifying **COVID as viral pneumonia** (0.02) is **higher** than classifying **viral pneumonia as COVID** (0.00)

```
[79] model.load_weights('./checkpoints/best_weights_InceptionV3Model.hdf5')
```

```
[80] metrics = model.evaluate(test_dataset,steps=50)
     print("Test accuracy:",metrics[1])

     50/50 [==============================] - 9s 178ms/step - loss: 0.1605 - accuracy: 0.9350
     Test accuracy: 0.9350000023841858
```



# Outcomes . . .

## Solving our problem:

Our model can be used in real life scenarios with its high accuracy, and scalability, and can be easily deployed onto web applications for healthcare professionals to use to accurately distinguish between Normal, Viral Pneumonia and COVID Cases, enabling faster diagnosis and treatment of viral and COVID-19 cases.

In the future, we can easily modify our model for other medical use cases (e.g detection of other lung infections like tuberculosis )

# Improvements and further Modifications ...

**Improving dataset:**

- Image Augmentation — Genetic Algorithm, Keras ImageDataGenerator
- Adopt oversampling
- Adding on more X-Ray images to classes with less data

**Other Pre-trained models:**

- VGG16, ResNet

**Other factors:**

- Symptoms of COVID and viral pneumonia
- Other forms of Medical Diagnosis Images (e.g Brain CT scans)

# What Have We Learned ...

## Exploratory Data Analysis

- Pixel Matrix analysis
- Eigenimages from Principal Component Analysis (PCA)
- Image Value Distributions

## Machine Learning

- Convolutional Neural Networks (CNN)
- Transfer Learning
  - Tensorflow and Keras
  - Pre-trained Model - InceptionV3
  - Optimisers - Adam Optimisation Algorithm
  - Checkpoints, Callbacks

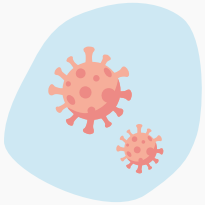
# THANK YOU

Stay safe &
remember to mask up indoors!

# What have we learned from this project? ...

## VENUS

Venus has a beautiful name and is the second planet from the Sun

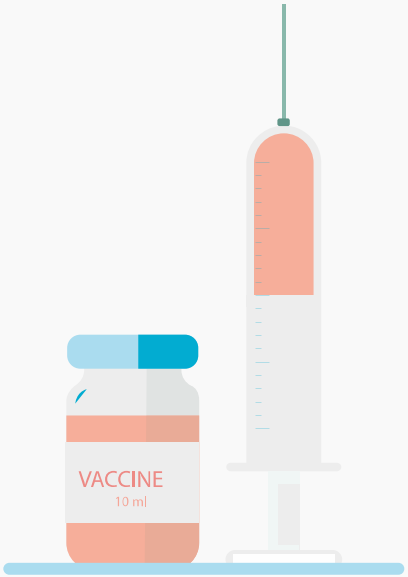


## MERCURY

Mercury is the closest planet to the Sun and the smallest one

# TABLE OF CONTENTS






# 01

# OBJECTIVES

You can enter a subtitle here if you need it

# INTRODUCTION

Mercury is the closest planet to the Sun and the smallest one in the Solar System

# BACKGROUND

## 01
### MERCURY
Mercury is the closest planet to the Sun

## 02
### MARS
Despite being red, Mars is actually a cold place

## 03
### JUPITER
It’s the biggest planet in the Solar System

## 04
### VENUS
Venus has a beautiful name, but it’s terribly hot

## 05
### SATURN
Saturn is the ringed one and a gas giant

## 06
### NEPTUNE
Neptune is the farthest planet from the Sun

# GOALS

## MERCURY

Mercury is the closest planet to the Sun and the smallest one in the Solar System

## VENUS

Venus has a beautiful name and is the second planet from the Sun. It’s terribly hot

## JUPITER

Jupiter is a gas giant and also the biggest planet in the entire Solar System

# METHODS

## NEPTUNE

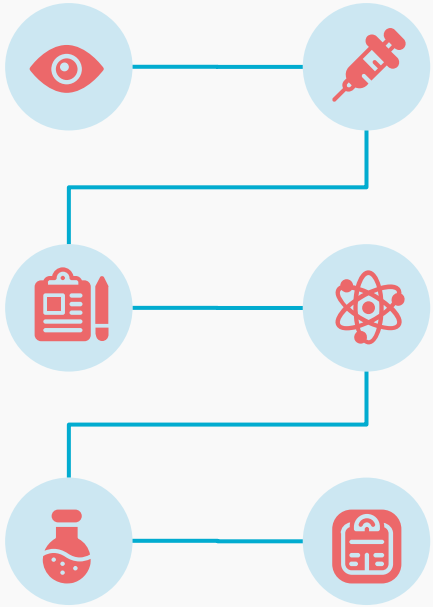
Neptune is the farthest planet from the Sun

## SATURN

Saturn is the ringed one and a gas giant

## VENUS

Venus has a beautiful name, but it’s terribly hot



## MERCURY

Mercury is the closest planet to the Sun

## MARS

Despite being red, Mars is a cold place

## JUPITER

It’s the biggest planet in the Solar System

# RESEARCH AND PUBLICATIONS

Venus has a beautiful name and is the second planet from the Sun

**by VENUS**

Mercury is the closest planet to the Sun and the smallest one in the Solar System

**by MERCURY**

Mars is full of iron oxide dust, which gives the planet its reddish cast

**by MARS**

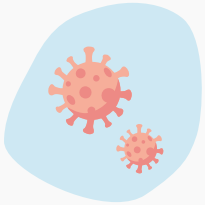
# RESEARCH RESOURCES

- Here you can list your reference websites or publications.
- Here you can list your reference websites or publications.
- Here you can list your reference websites or publications.
- Here you can list your reference websites or publications.
- Here you can list your reference websites or publications.
- Here you can list your reference websites or publications.
- Here you can list your reference websites or publications.

# FACTORS TO CONSIDER

## VENUS

Venus has a beautiful name and is the second planet from the Sun

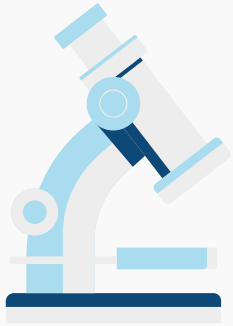


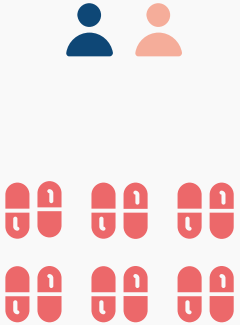
## MERCURY

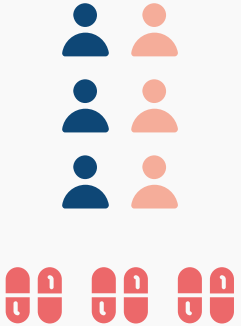
Mercury is the closest planet to the Sun and the smallest one

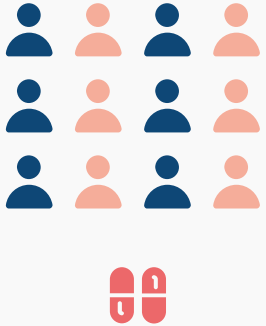
# CLINICAL TRIAL

| PRECLINICAL | PHASE 1 | PHASE 2 | PHASE 3 |
|---|---|---|---|
| Lab Studies | Human Safety | Expanded Safety | Efficacy & Safety |









# TRIAL TIMELINE

## RESEARCH

Venus is the second planet from the Sun

## EXPERIMENTATION

Despite being red, Mars is a cold place

## CONCLUSIONS

Mercury is the closest planet to the Sun

## PRECLINICAL

Jupiter is a gas giant and the biggest planet

## RESULTS

Saturn is a gas giant and has several rings

# PHASE 1

## SAMPLE

Young healthy people

20 - 80 Participants

## SECONDARY EFFECTS

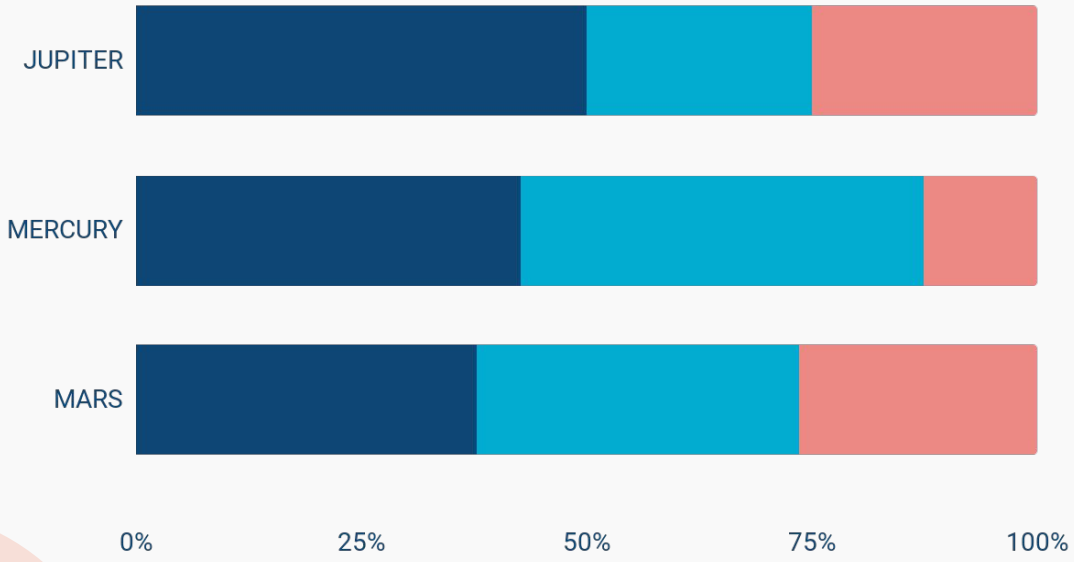
**80%**

**Success rate**

Mercury is the closest planet to the Sun

| RESULTS | GOAL | 6 MONTHS |
| --- | --- | --- |
| Venus is the second planet from the Sun | Study the safety of medication | Experimentation time |

# TENDENCY




## MARS

Despite being red, Mars is a cold place

## JUPITER

It’s the biggest planet in the Solar System

## MERCURY

Mercury is the closest planet to the Sun

To modify this graph, click on it, follow the link, change the data and paste the resulting graph here

# A PICTURE IS WORTH A THOUSAND WORDS

# RESULTS

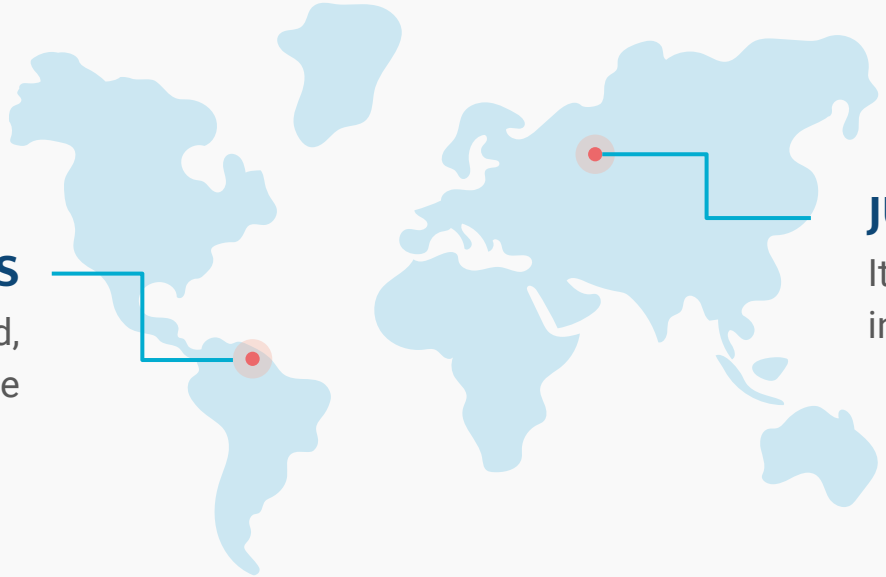
## Experiment A

| | OUTCOME | | |
|---|---|---|---|
| **TREATMENT** | Test 1 | Test 2 | Test 3 |
| **Group 1** | 315 | 285 | 600 |
| **Group 2** | 210 | 390 | 600 |
| **Group 3** | 240 | 165 | 580 |

## Experiment B

| | OUTCOME | | |
|---|---|---|---|
| **TREATMENT** | Test 1 | Test 2 | Test 3 |
| **Group 1** | 189 | 285 | 474 |
| **Group 2** | 210 | 234 | 444 |
| **Group 3** | 367 | 123 | 396 |

# RESULTS



### MARS

Despite being red,
Mars is a cold place

### JUPITER

It’s the biggest planet
in the Solar System

# RESULTS ANALYSIS

**MARS**

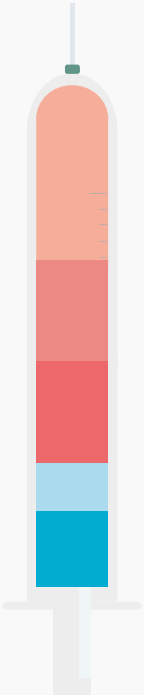
It’s actually a cold place

**JUPITER**

It’s the biggest planet

**MERCURY**

It’s a small planet



81% **MERCURY**

54% **MARS**

42% **JUPITER**

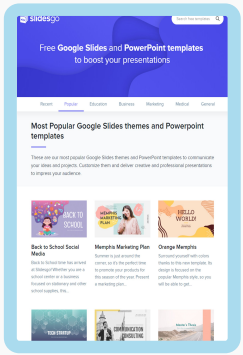
23% **VENUS**

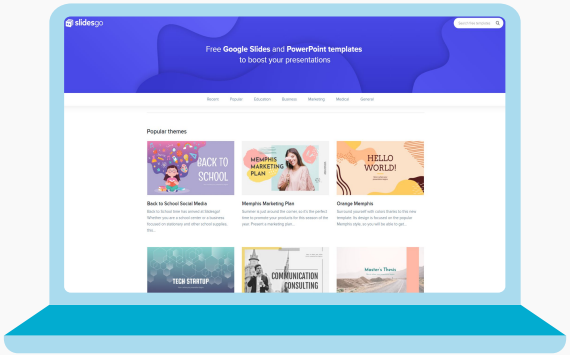
12% **NEPTUNE**

# 10,000,000

Big numbers catch your audience’s attention

# MULTIMEDIA

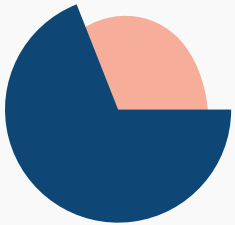



## DESKTOP SOFTWARE

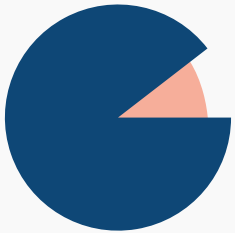
You can replace the image on the screen with your own work

# SUCCESS RATE



75%

Patients cured



97%

Secondary effects

# CONCLUSION

Mercury is the closest planet to the Sun and the smallest one in the Solar System—it's only a bit larger than the Moon

# ALTERNATIVE RESOURCES

# RESOURCES

**VECTOR**

- Emergency ambulance van and person in hazmat suit
- Coronavirus vaccine development
- Coronavirus vaccine development concept
- Virus cure concept
- Mental health awareness and meditation concept
- Character wearing protection and holding a covid-19 test
- Science team trying to develop coronavirus cure
- Web responsive design

**PHOTO**

- Heart and medical dust mask copy space
- Front view of coronavirus concept with medical mask

# Instructions for use

In order to use this template, you must credit **Slidesgo** by keeping the **Thanks** slide.

**You are allowed to:**

- Modify this template.
- Use it for both personal and commercial projects.

**You are not allowed to:**

- Sublicense, sell or rent any of Slidesgo Content (or a modified version of Slidesgo Content).
- Distribute Slidesgo Content unless it has been expressly authorized by Slidesgo.
- Include Slidesgo Content in an online or offline database or file.
- Offer Slidesgo templates (or modified versions of Slidesgo templates) for download.
- Acquire the copyright of Slidesgo Content.

For more information about editing slides, please read our FAQs or visit Slidesgo School:

https://slidesgo.com/faqs and https://slidesgo.com/slidesgo-school

# Fonts & colors used

This presentation has been made using the following fonts:

**Mukta**
(https://fonts.google.com/specimen/Mukta)

**Roboto**
(https://fonts.google.com/specimen/Roboto)

#0e4776 #02acd0 #aadbee #f5ad9a #ed8984 #ec686a

# Stories by Freepik

Create your Story with our illustrated concepts. Choose the style you like the most, edit its colors, pick the background and layers you want to show and bring them to life with the animator panel! It will boost your presentation. Check out How it Works.

Pana

Amico

Bro

Rafiki

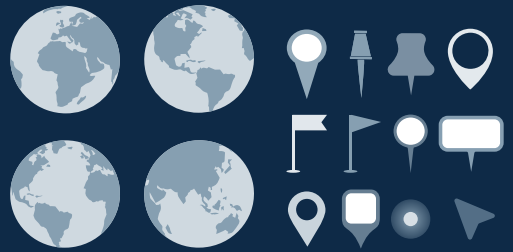
Cuate

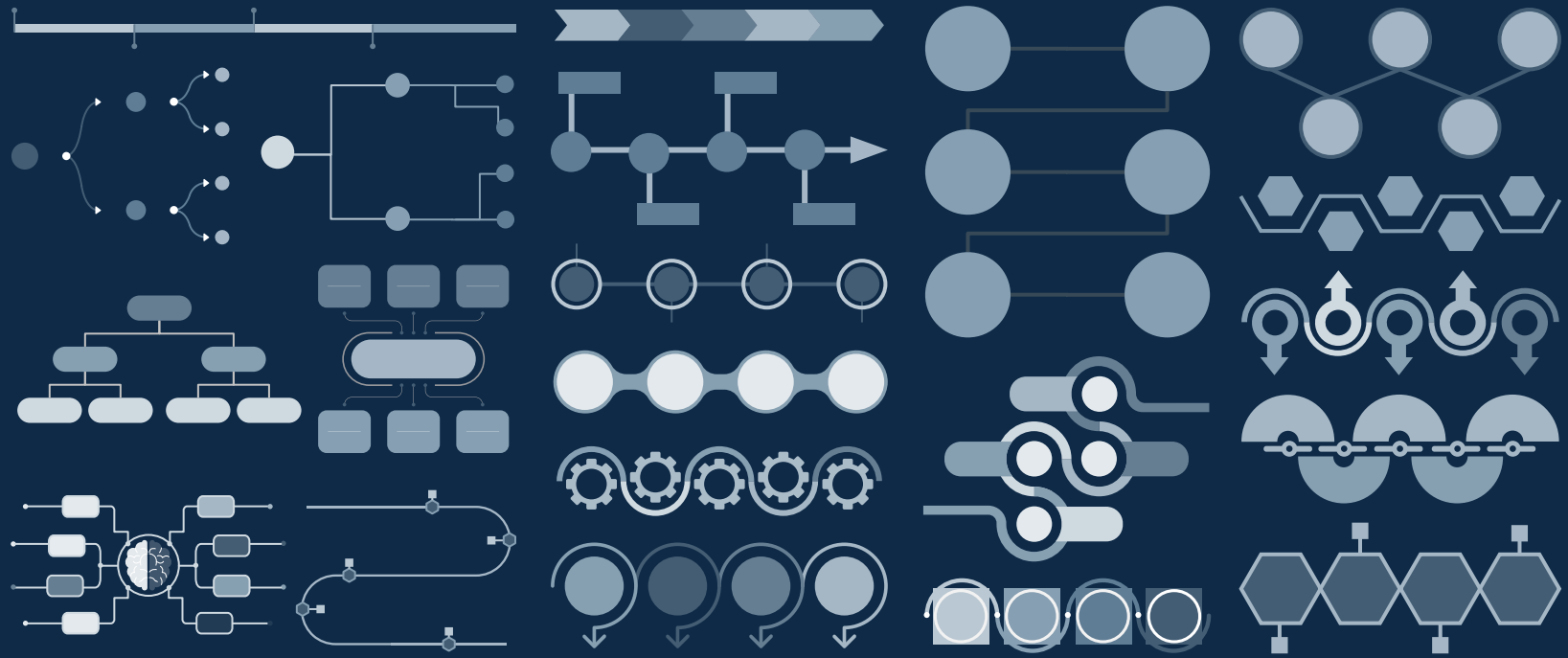
# Use our editable graphic resources...

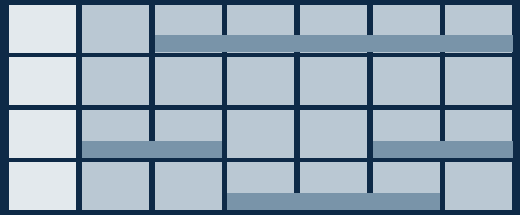
You can easily resize these resources without losing quality. To change the color, just ungroup the resource and click on the object you want to change. Then, click on the paint bucket and select the color you want. Group the resource again when you're done. You can also look for more infographics on Slidesgo.

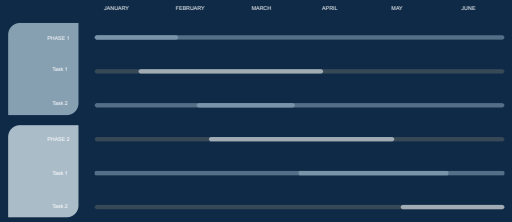


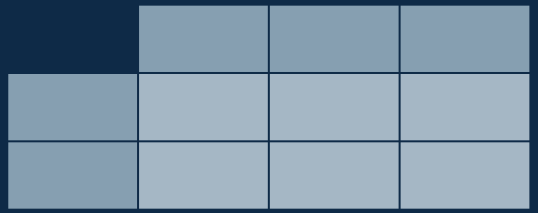


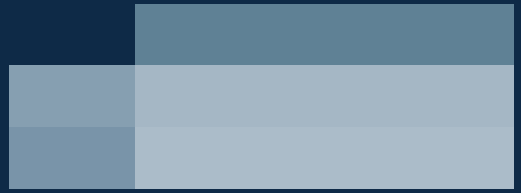



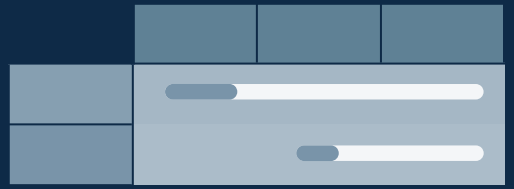


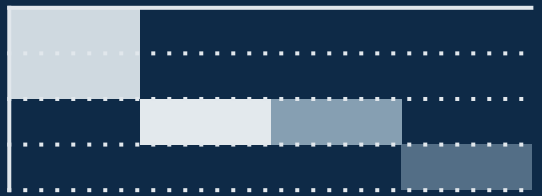


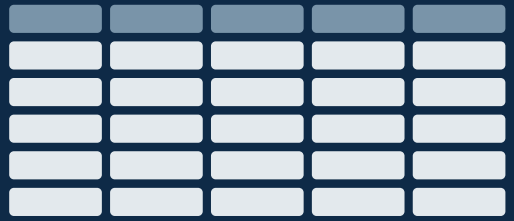

JANUARY
FEBRUARY
MARCH
APRIL
MAY
JUNE
PHASE 1
Task 1
Task 2
PHASE 2
Task 1
Task 2

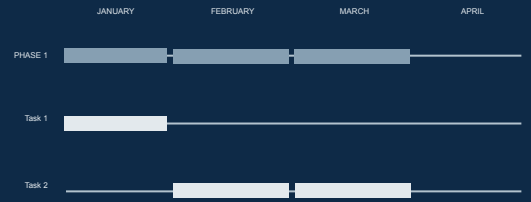


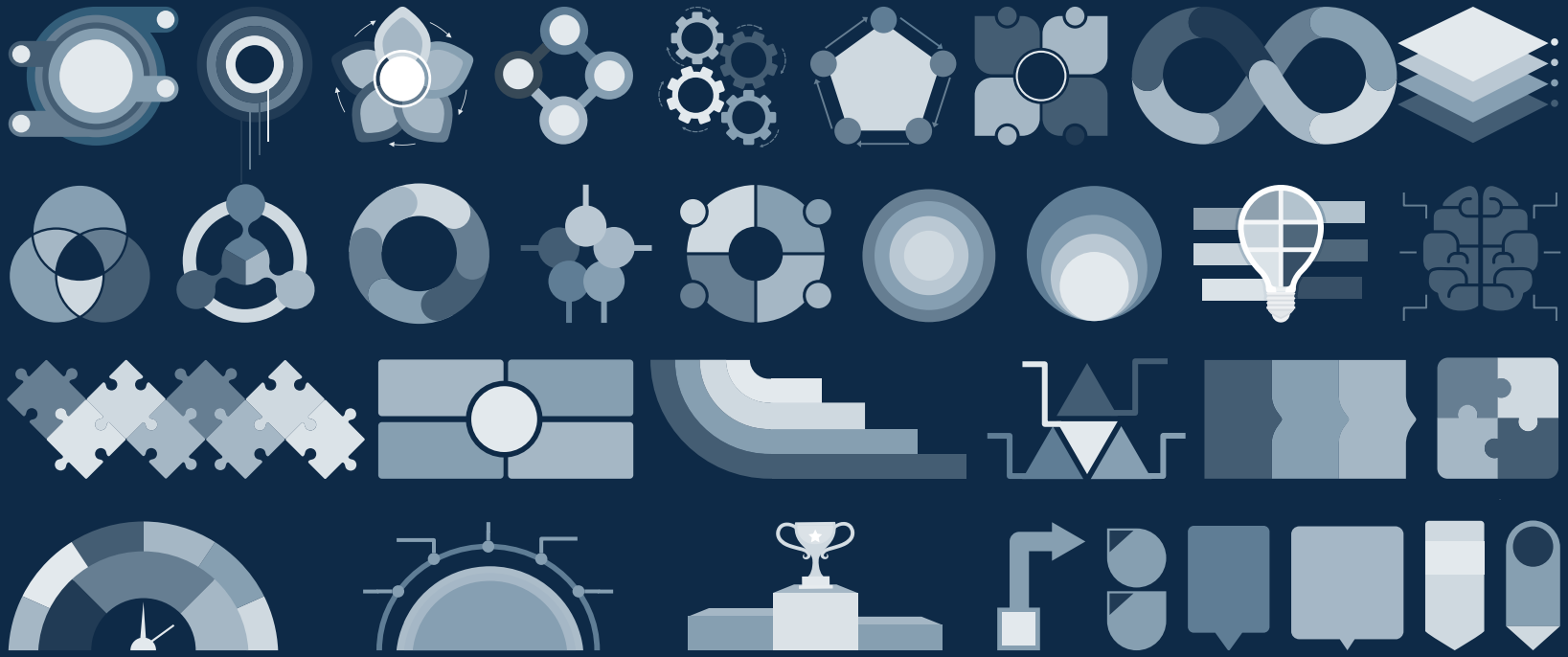


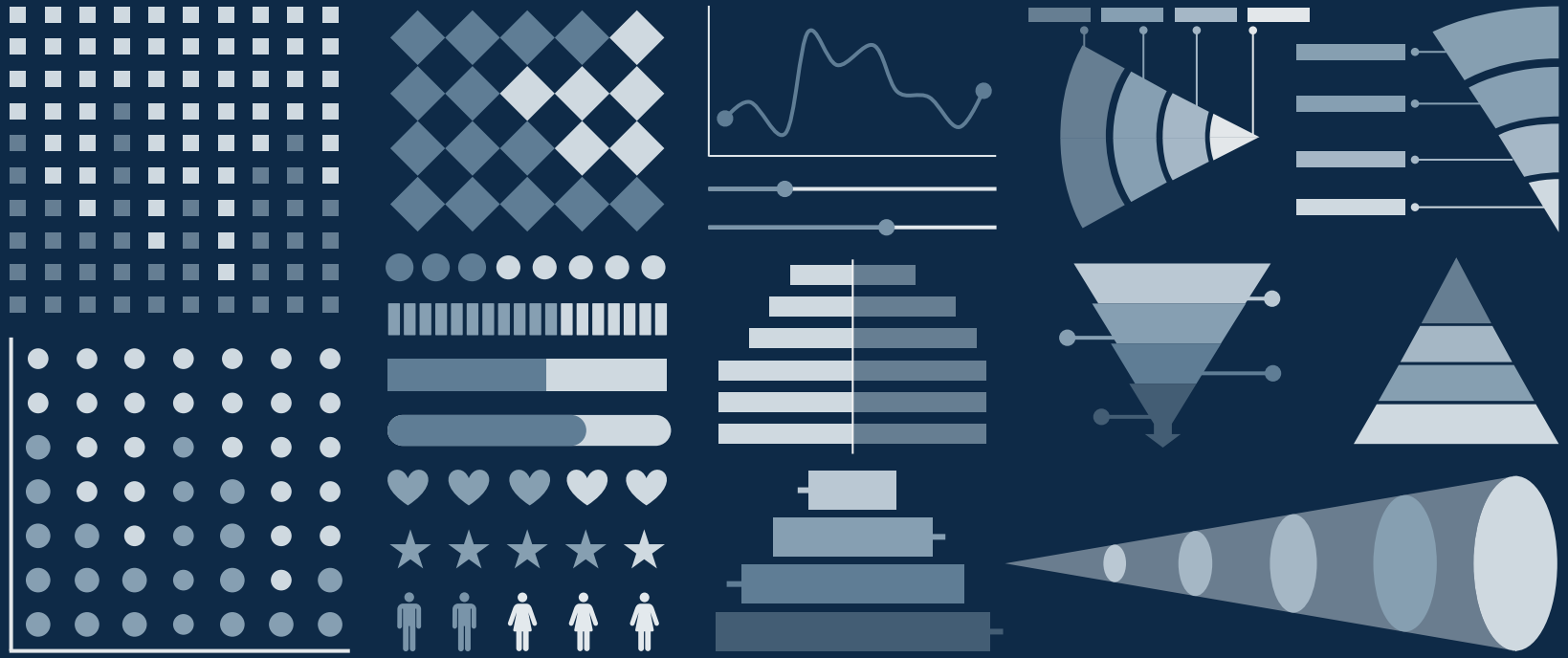







JANUARY
FEBRUARY
MARCH
APRIL
PHASE 1
Task 1
Task 2





# ...and our sets of editable icons

You can resize these icons without losing quality.
You can change the stroke and fill color; just select the icon and click on the paint bucket/pen.
In Google Slides, you can also use Flaticon's extension, allowing you to customize and add even more icons.

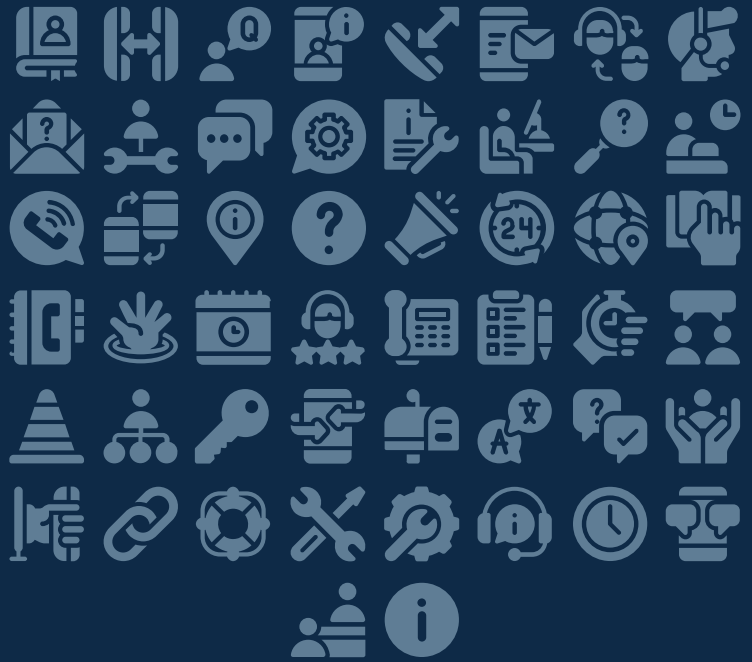
## Educational Icons

## Medical Icons

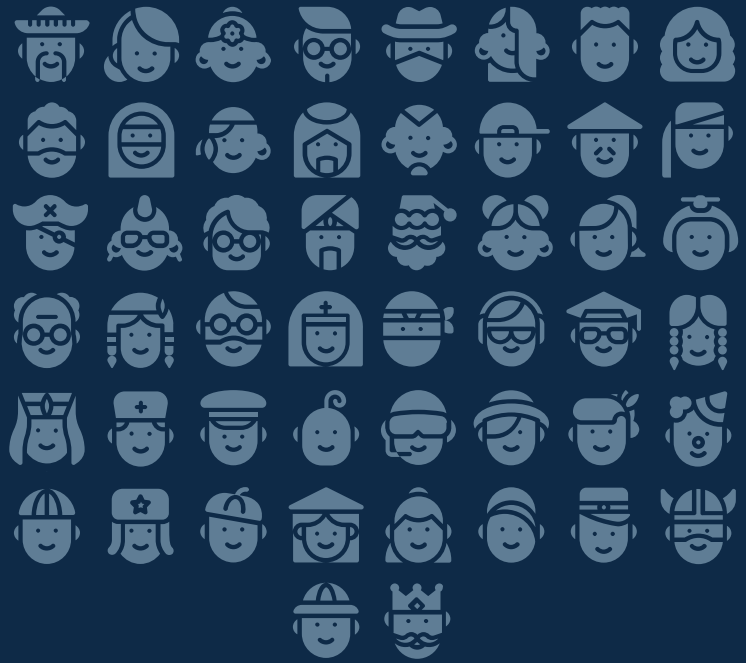
## Business Icons

## Teamwork Icons

## Help & Support Icons



## Avatar Icons



## Creative Process Icons


## Performing Arts Icons

# Nature Icons

# SEO & Marketing Icons


slidesgo