



# Individual Study III

## Tuberculosis Chest X-rays



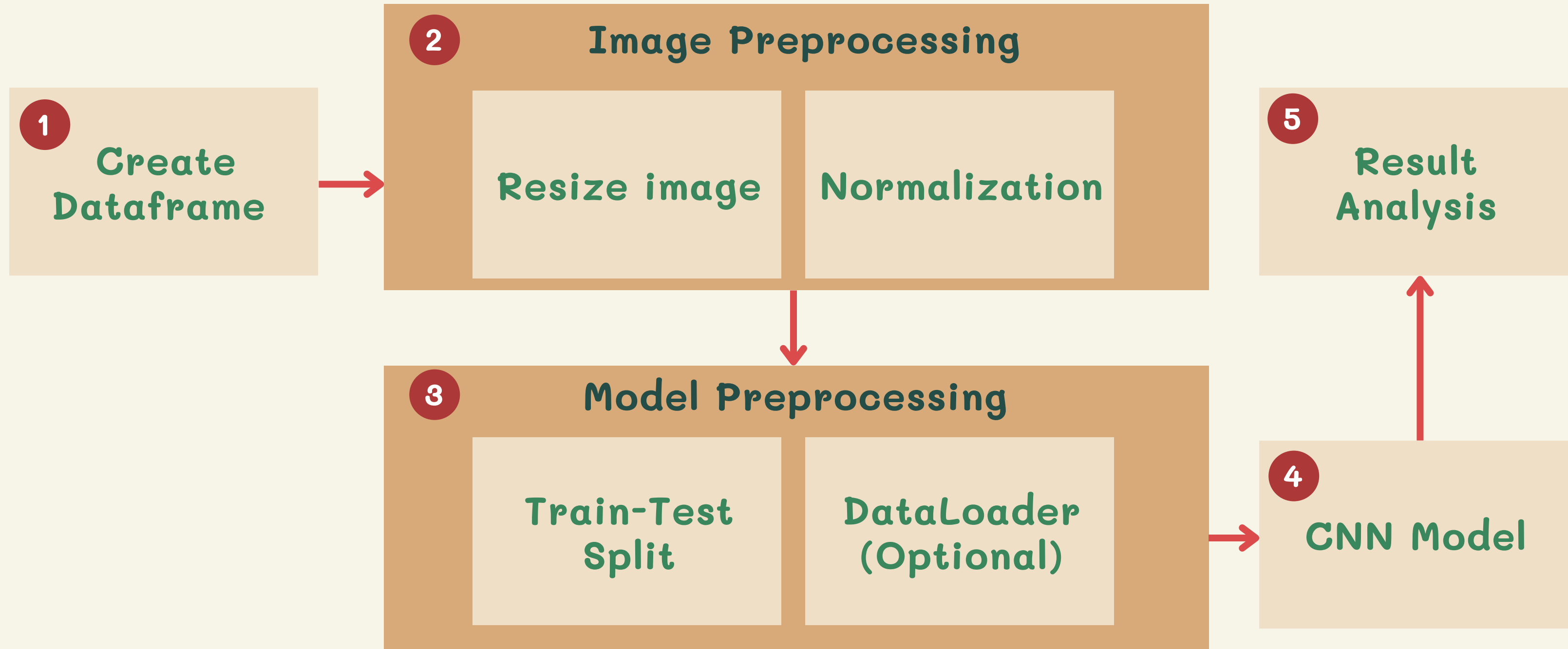
Peeranath Theerawatanachai 6231343821



# Objective

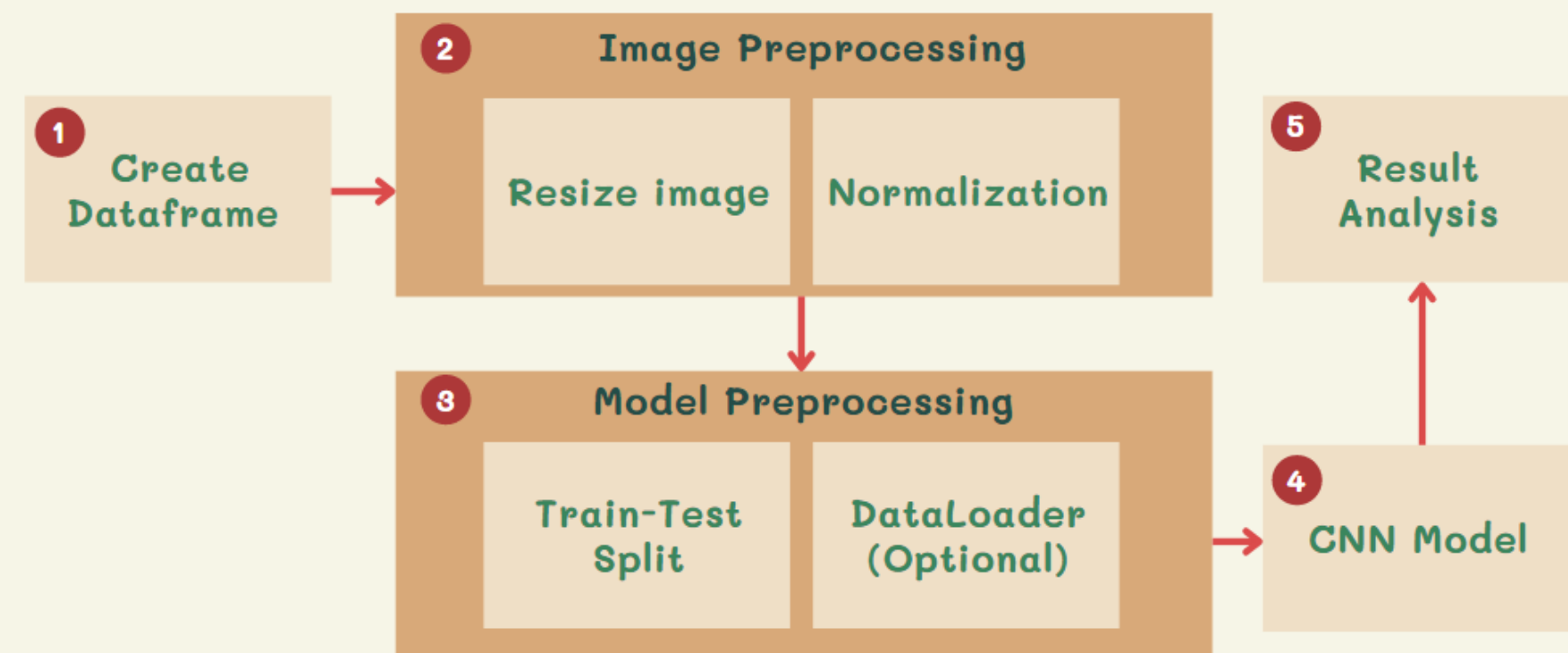
1. To predict Tuberculosis from Chest X-ray images
2. To study about Machine Learning (Deep learning from computer vision) which involves Convolutional Neural Network

# Architectures

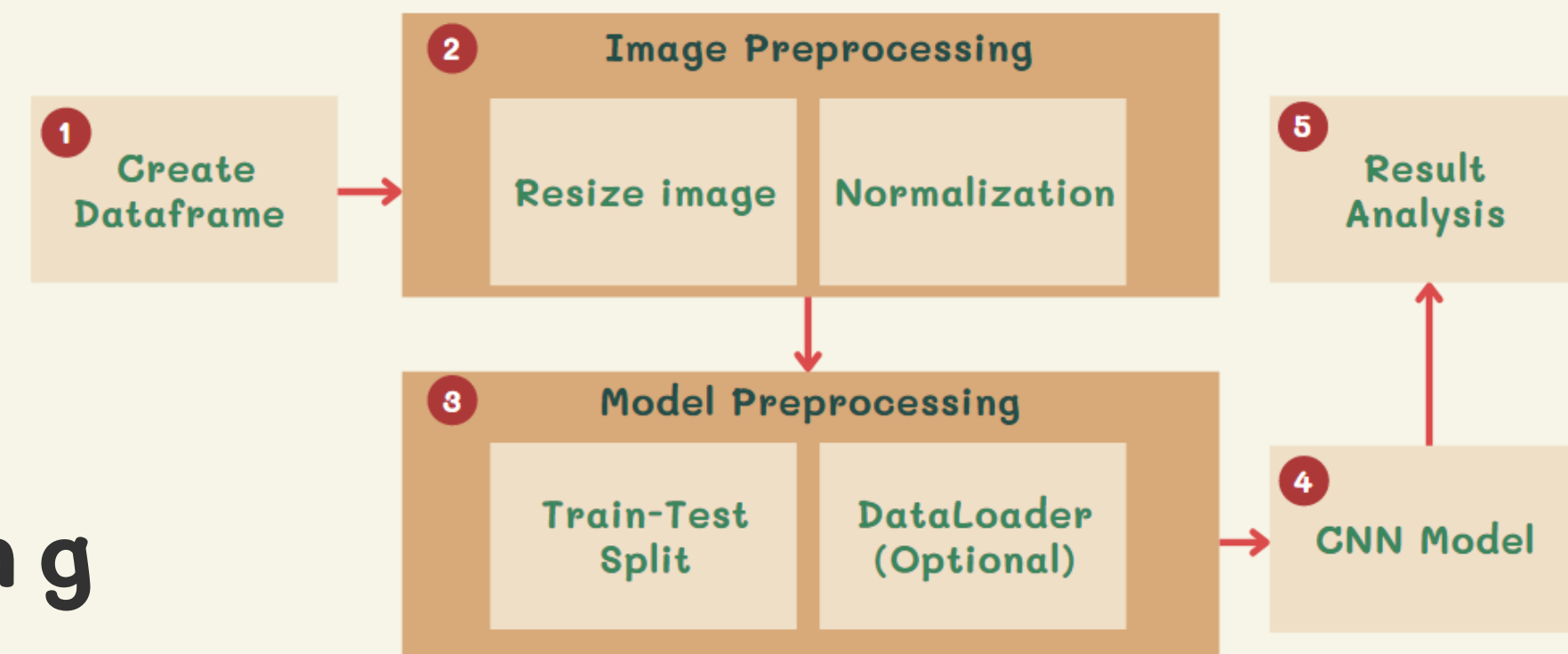


# 1. Create dataframe

- Download data from Kaggle
- Create dataframe from shenzhen\_metadata.csv by adding column label from column finding where 0 means normal and 1 means abnormal (Tuberculosis)

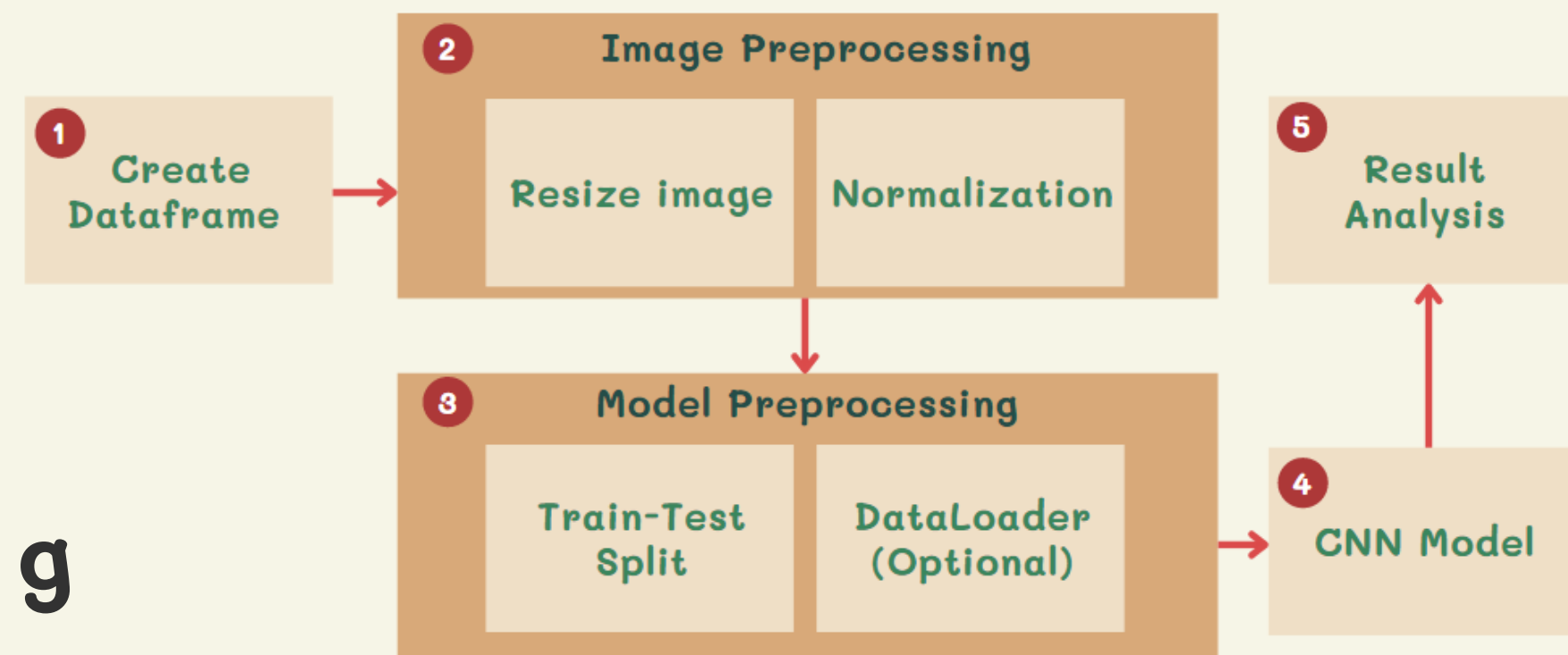


## 2. Image Preprocessing



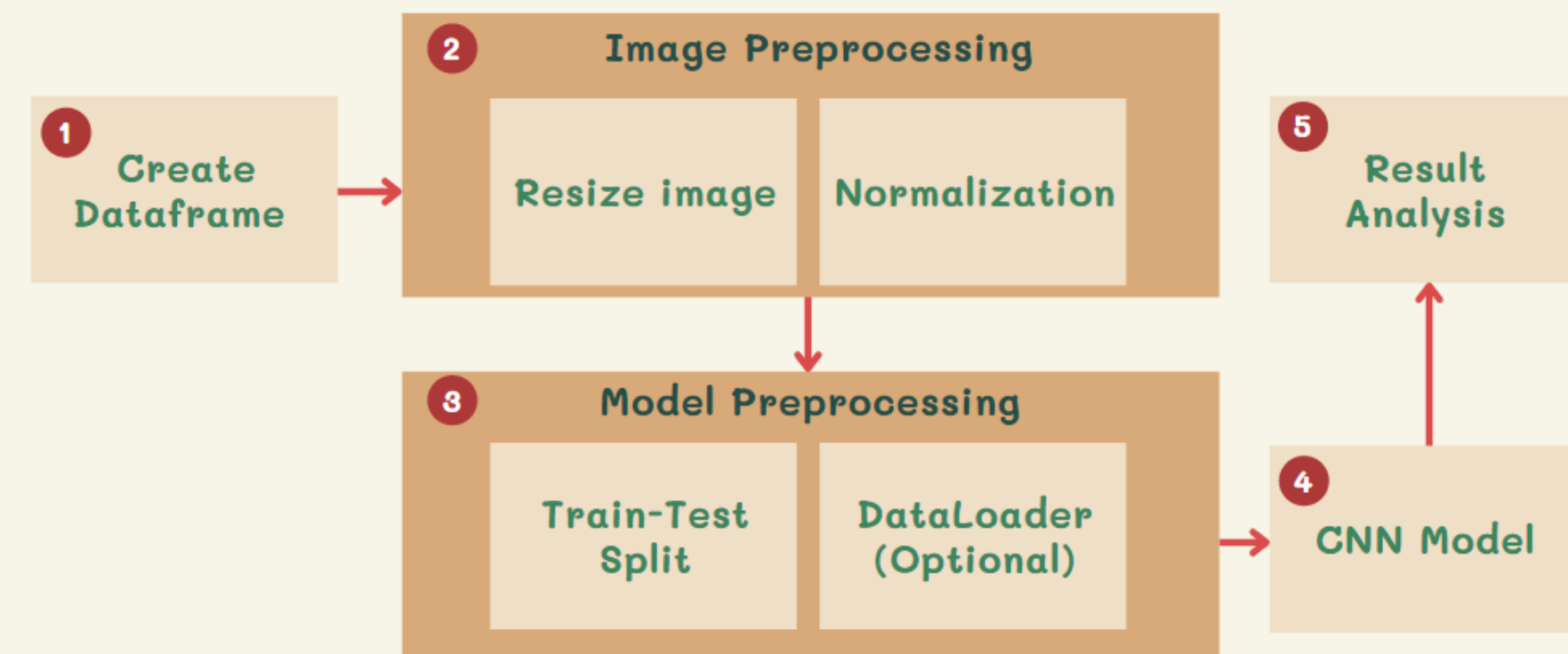
- **Resize** Chest X-ray images to 256\*256 and 224\*224 pixels
- **Normalize** each image by dividing every pixel by 255, then subtract with mean of all images, and divide by standard deviance of all images to make each image has better intensity value distribution

### 3. Model Preprocessing



- Use **Train-test split** with split size = 15% to divide images for training (562 images) and validating (100 images)
- Create Class XRayDataset to keep image intensity level and labels
- Use **DataLoader** to divide batch for training and validating by using batch size = 16 for training set and batch size = 32 for validating set

## \* 4-5. CNN Model & Result analysis



### Model 1

- (0): **Conv2d**(1, 4, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))
- (1): **BatchNorm2d**(4, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)
- (2): **ReLU**(inplace=True)
- (3): **MaxPool2d**(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)
- (4): **Conv2d**(4, 4, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))
- (5): **BatchNorm2d**(4, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)
- (6): **ReLU**(inplace=True)
- (7): **MaxPool2d**(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)
- (8): **Linear**(in\_features=16384, out\_features=2, bias=True)



## \* 4-5. CNN Model & Result analysis

### Training 1 (Overfit)

#### Parameter:

optimizer = Adam

Learning rate = 0.00004

criterion = CrossEntropyLoss

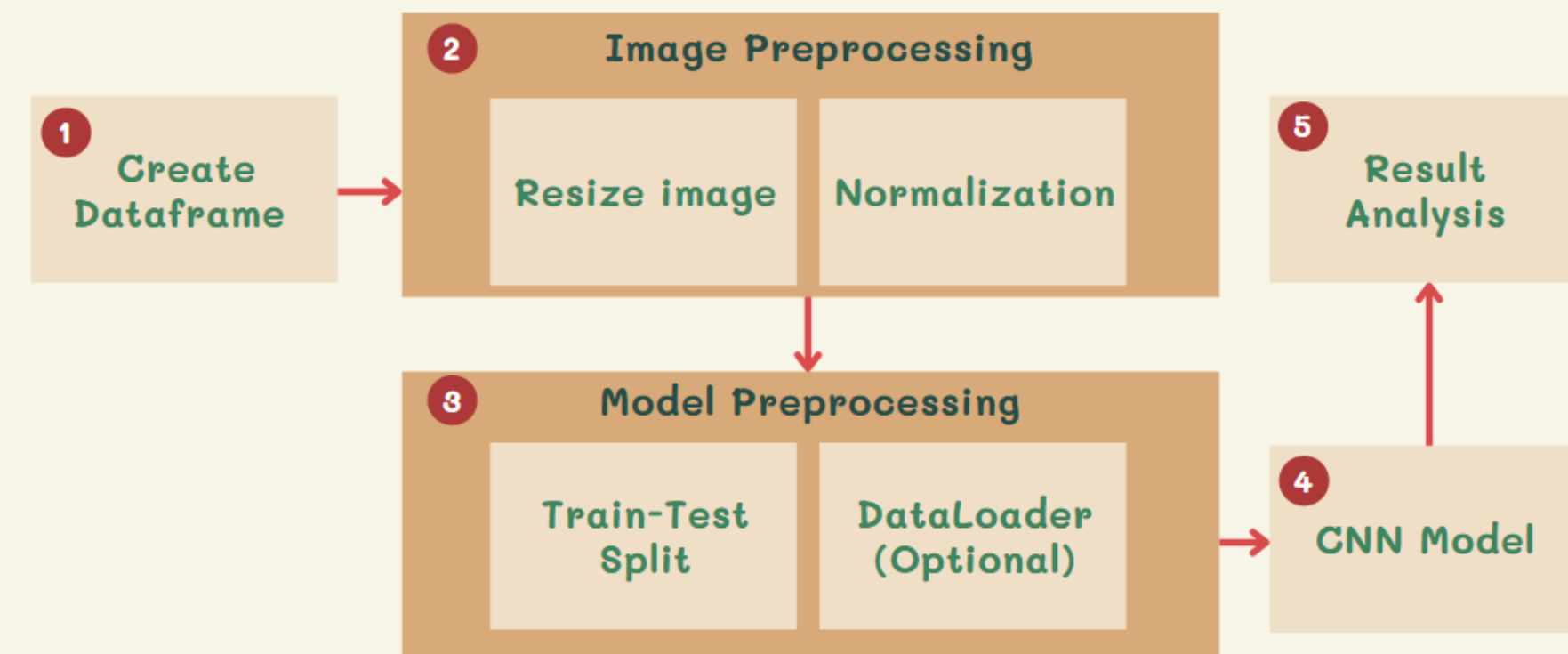
scheduler = ReduceLROnPlateau

number of epochs = 400

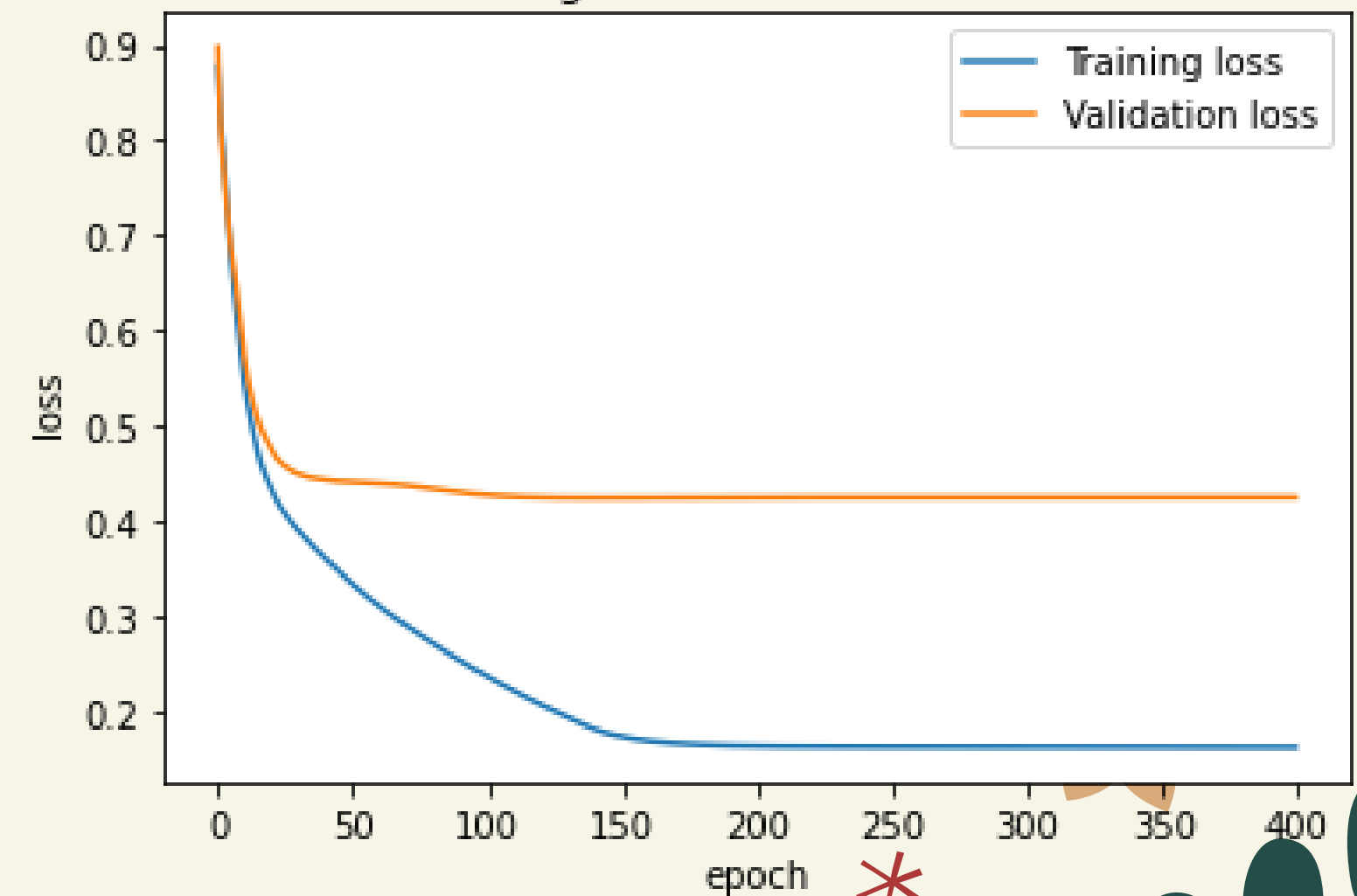
#### Accuracy:

Training accuracy: 96.09%

Validate accuracy: 84%



Training loss and Validation loss





## \* 4-5. CNN Model & Result analysis

### *Training 2: Add Dropout = 0.2*

#### **Parameter:**

optimizer = Adam

Learning rate = 0.00004

criterion = CrossEntropyLoss

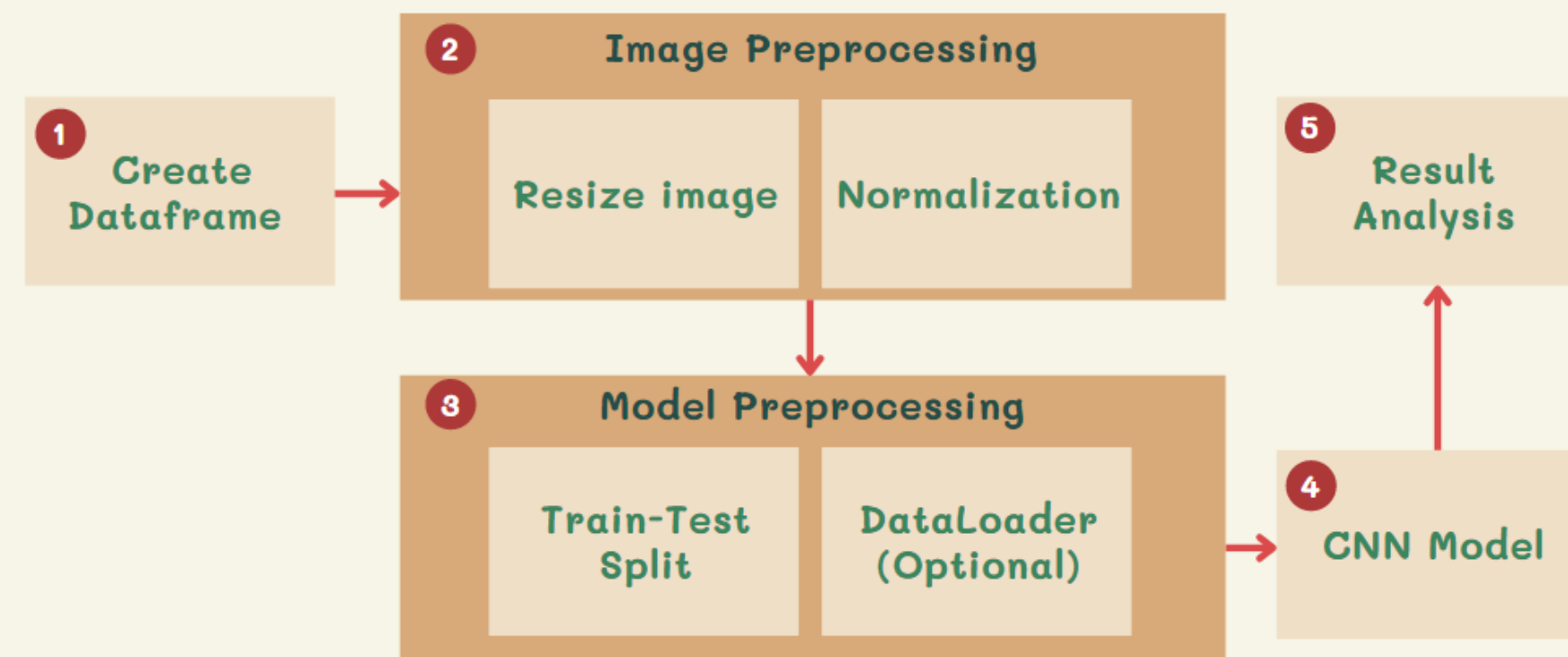
scheduler = ReduceLROnPlateau

number of epochs = 200

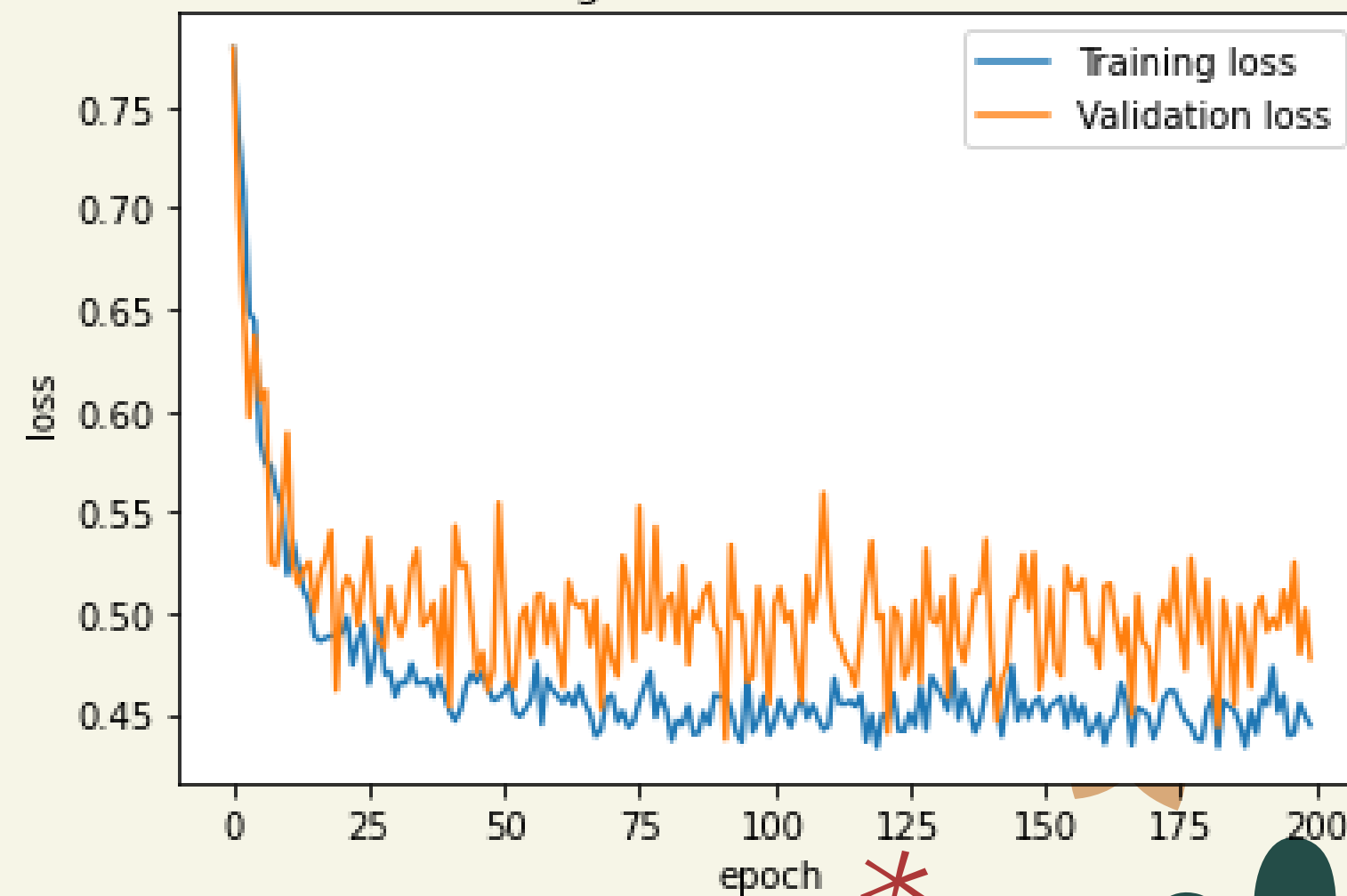
#### **Accuracy:**

Training accuracy: 80.96%

Validate accuracy: 79%



Training loss and Validation loss



## \* 4-5. CNN Model & Result analysis

### *Training 3: Using DataLoader + Dropout = 0.2*

#### **Parameter:**

optimizer = Adam

Learning rate = 0.000004

criterion = CrossEntropyLoss

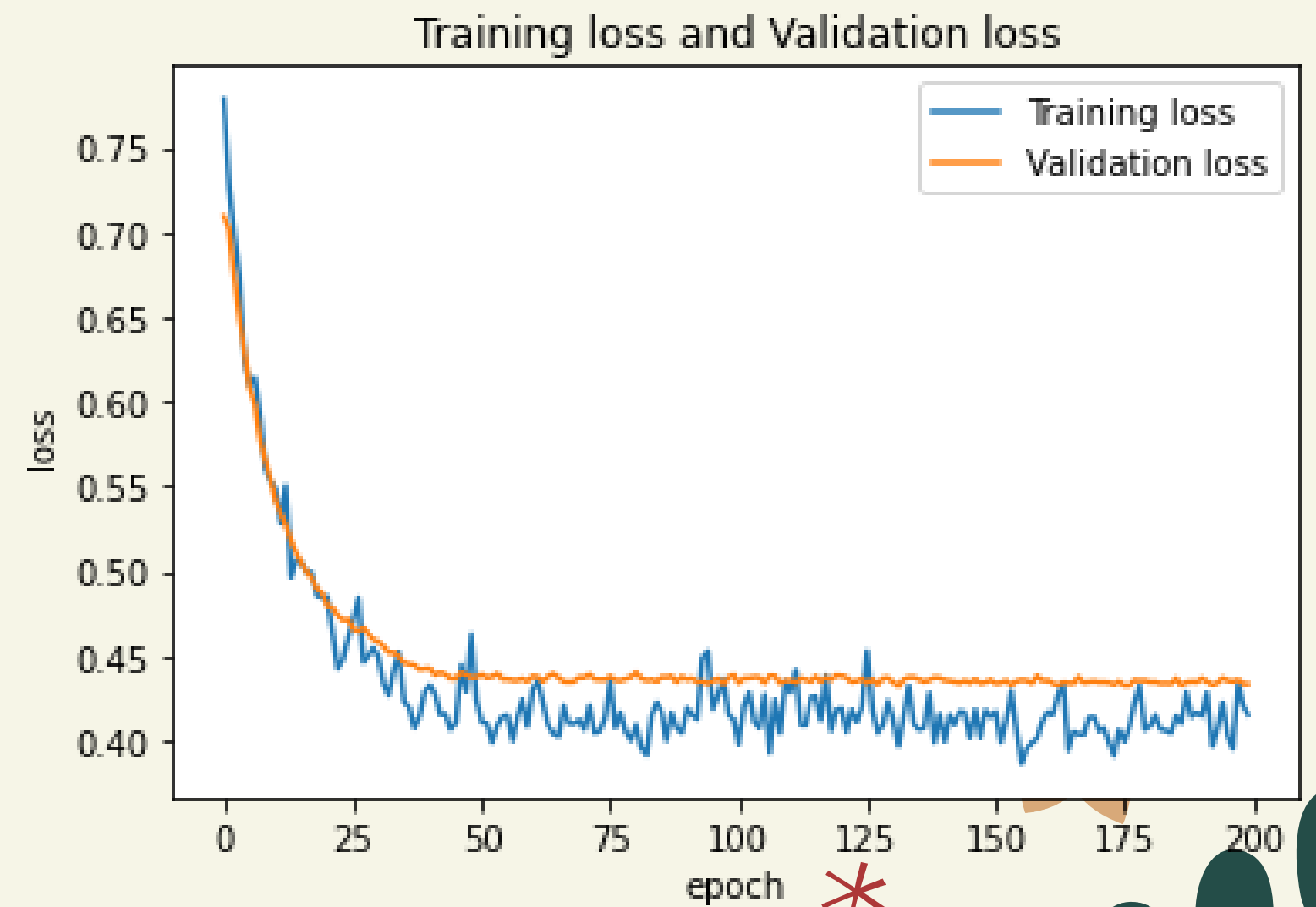
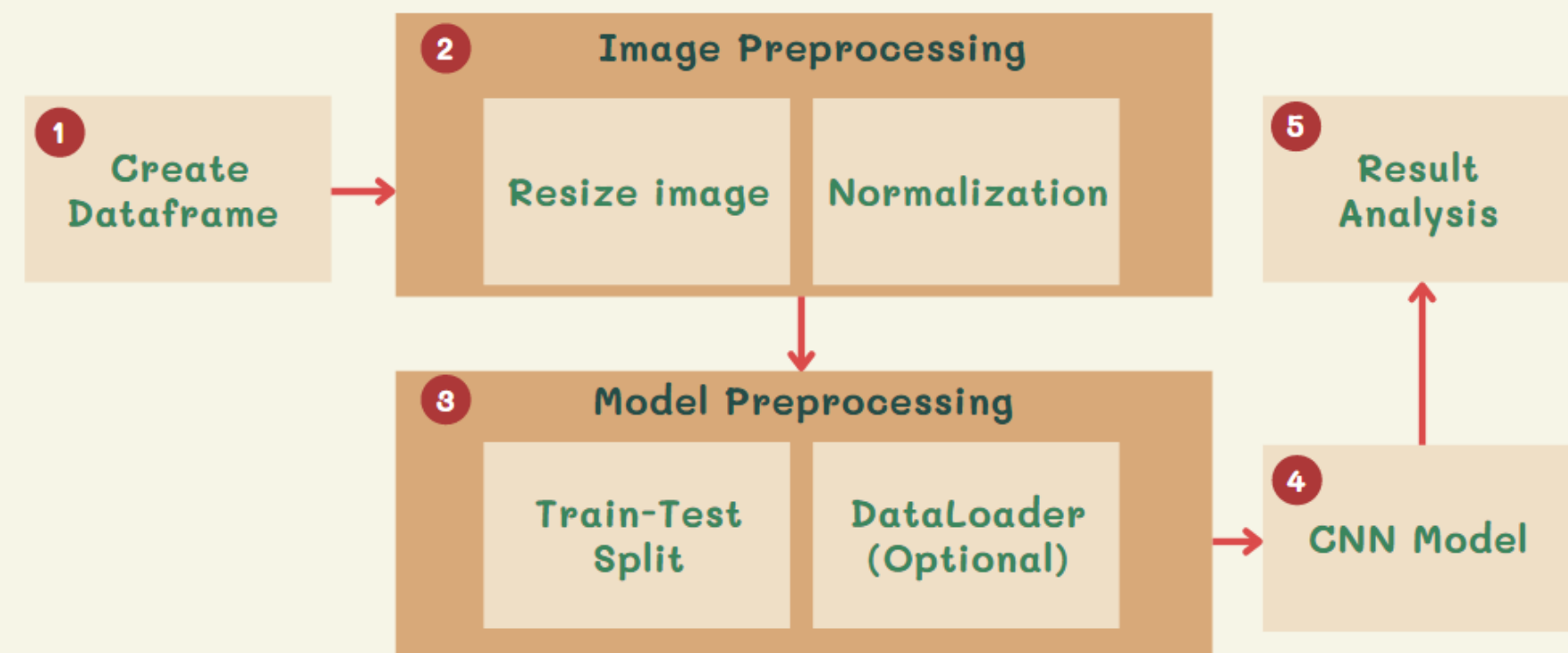
scheduler = ReduceLROnPlateau

number of epochs = 200

#### **Accuracy:**

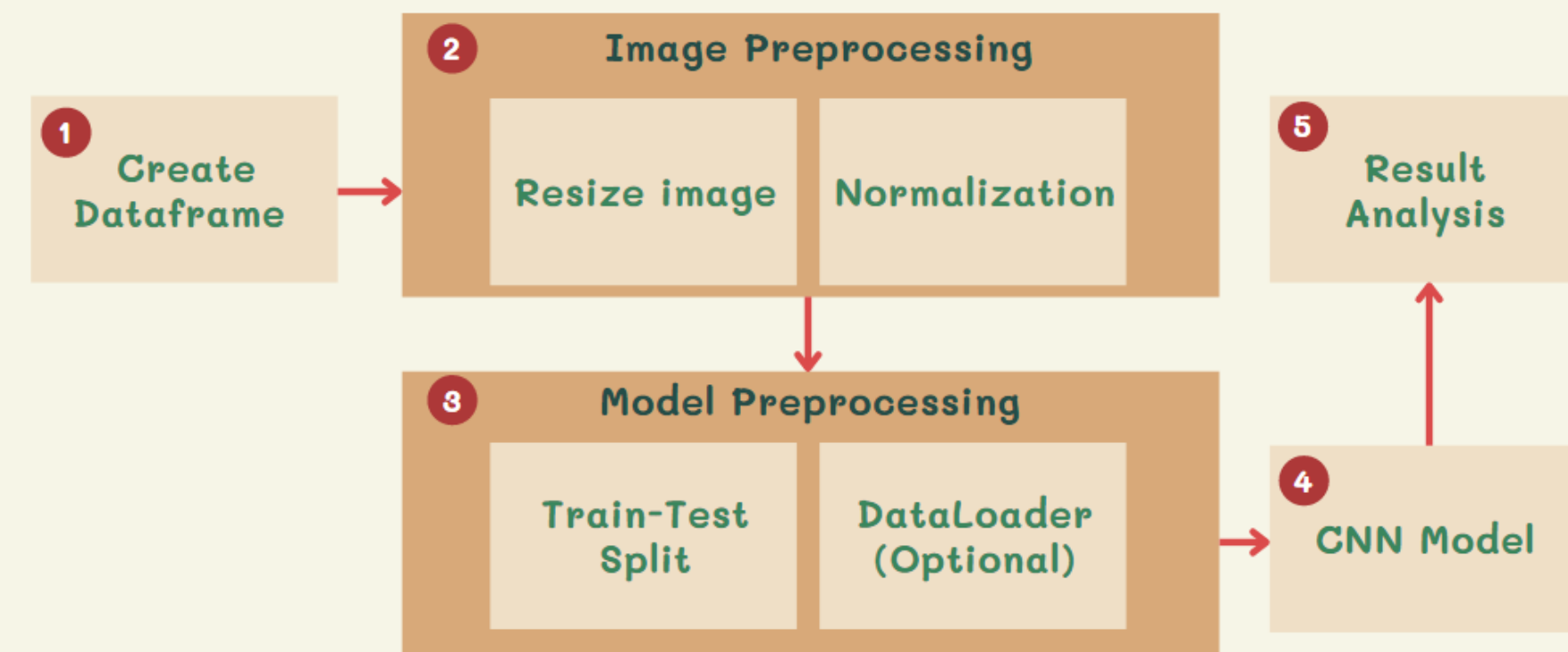
Training accuracy: 84.55%

Validate accuracy: 86.72%





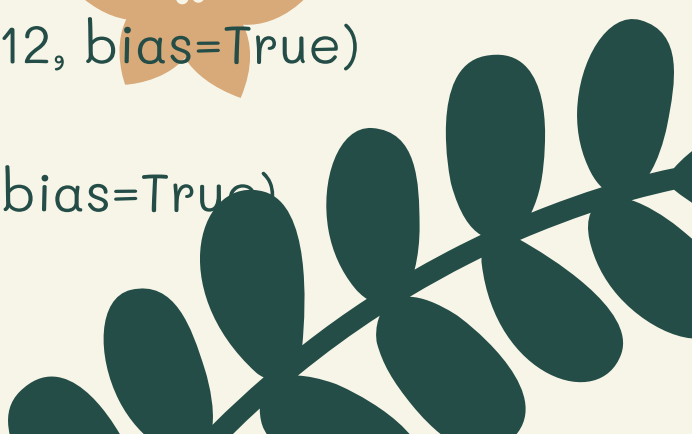
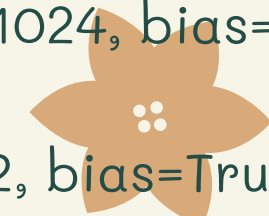
## \* 4-5. CNN Model & Result analysis



### Model 2 (Using DataLoader)

(0): **Conv2d**(1, 32, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  
(1): **ReLU()**  
(2): **Conv2d**(32, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  
(3): **ReLU()**  
(4): **MaxPool2d**(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)  
(5): **Conv2d**(64, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  
(6): **ReLU()**  
(7): **Conv2d**(128, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  
(8): **ReLU()**  
(9): **MaxPool2d**(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)

(10): **Conv2d**(128, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  
(11): **ReLU()**  
(12): **Conv2d**(256, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  
(13): **ReLU()**  
(14): **MaxPool2d**(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)  
(15): **Flatten**(start\_dim=1, end\_dim=-1)  
(16): **Linear**(in\_features=262144, out\_features=1024, bias=True)  
(17): **ReLU()**  
(18): **Linear**(in\_features=1024, out\_features=512, bias=True)  
(19): **ReLU()**  
(20): **Linear**(in\_features=512, out\_features=2, bias=True)



## \* 4-5. CNN Model & Result analysis

### Training 1

#### Parameter:

optimizer = Adam

Learning rate = 0.00001

criterion = CrossEntropyLoss

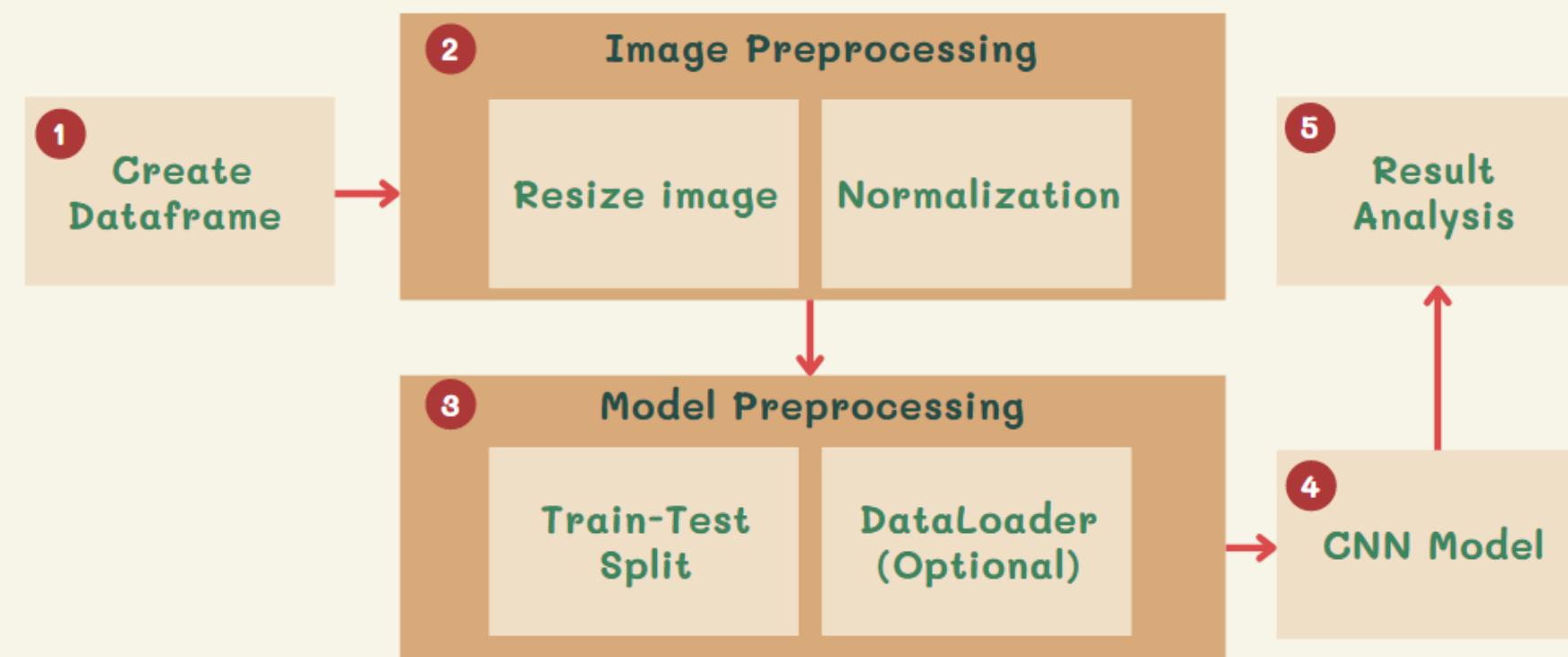
scheduler = ReduceLROnPlateau

number of epochs = 100

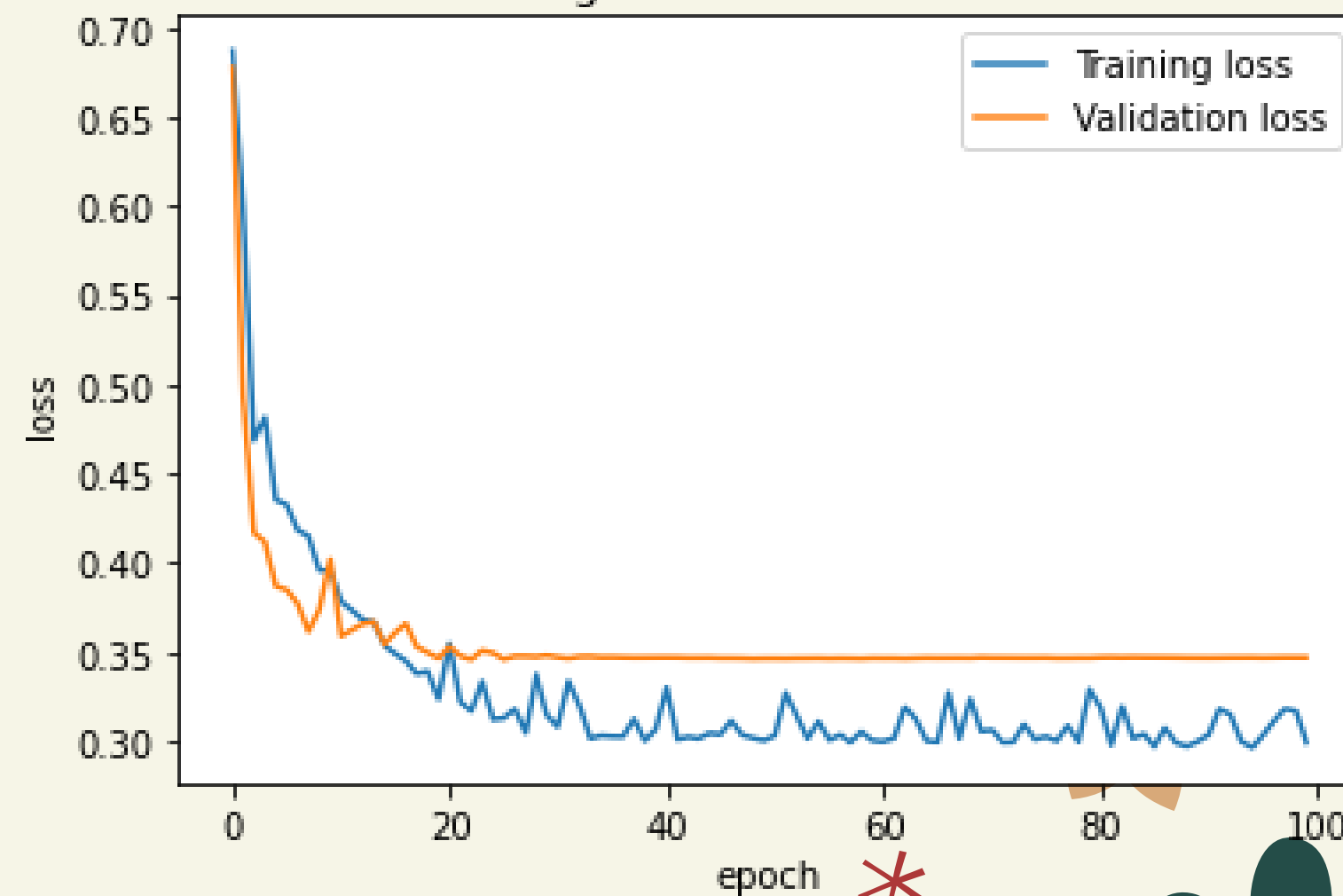
#### Accuracy:

Training accuracy: 87.85%

Validate accuracy: 85.16%



Training loss and Validation loss



## \* 4-5. CNN Model & Result analysis

### Training 2: Add Dropout = 0.3

#### Parameter:

optimizer = Adam

Learning rate = 0.00001

criterion = CrossEntropyLoss

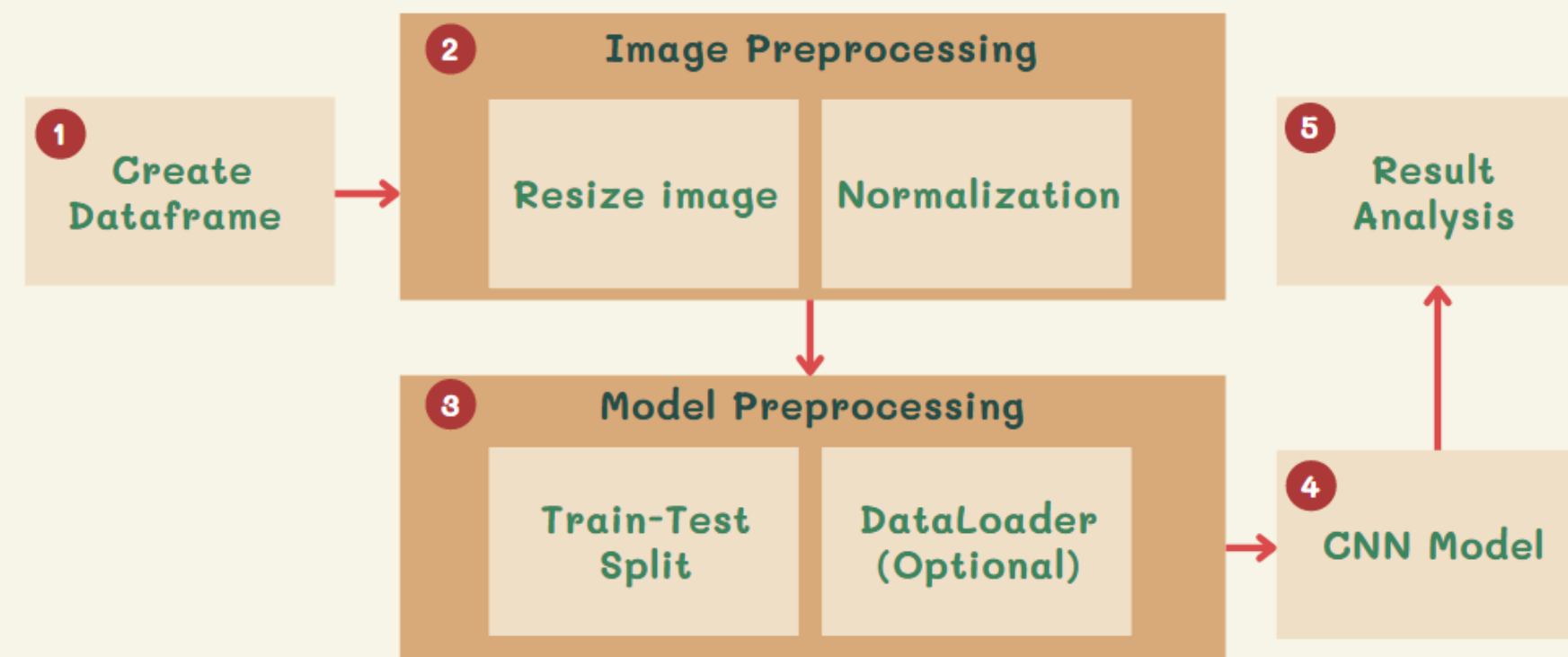
scheduler = ReduceLROnPlateau

number of epochs = 100

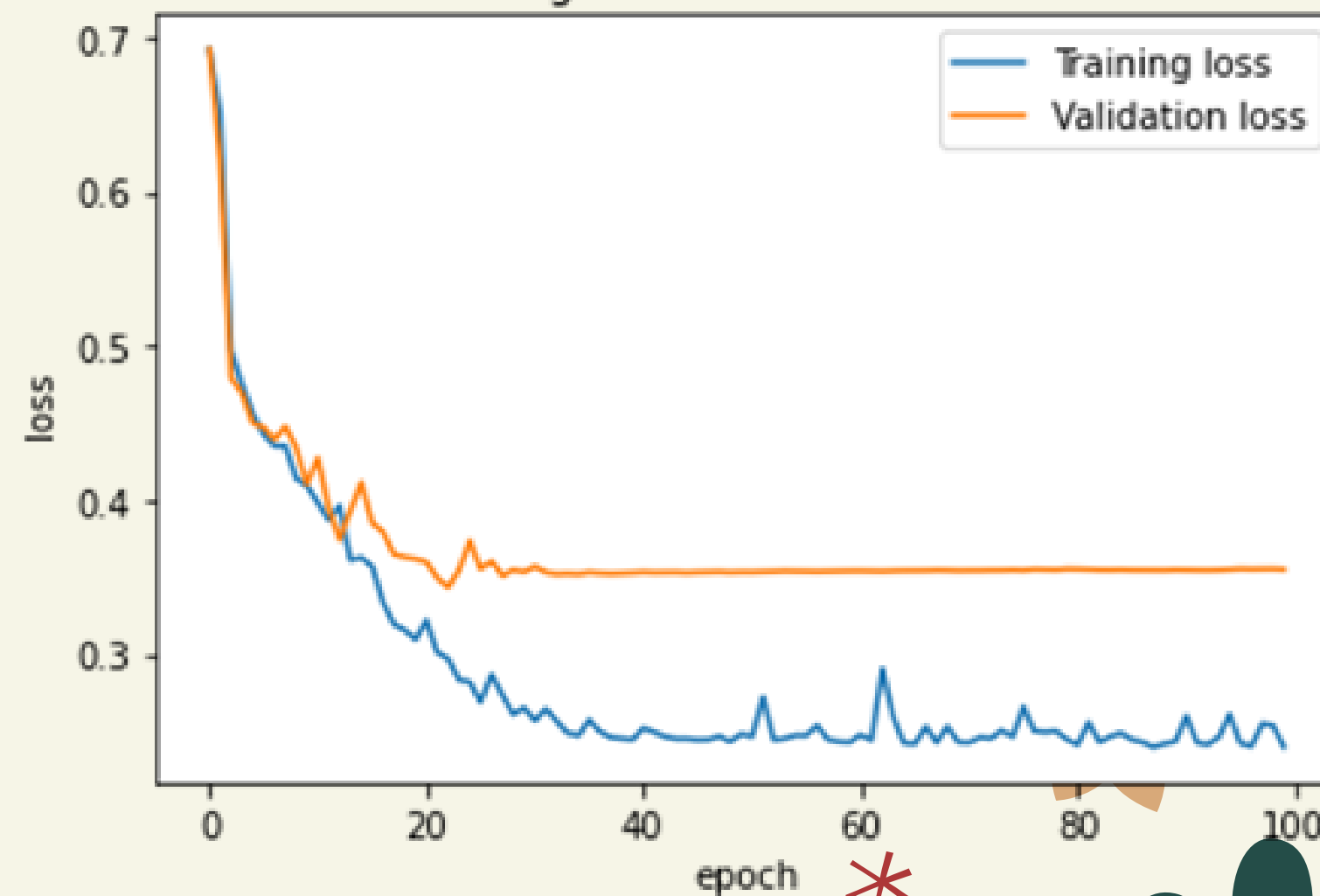
#### Accuracy:

Training accuracy: 88.89%

Validate accuracy: 88%



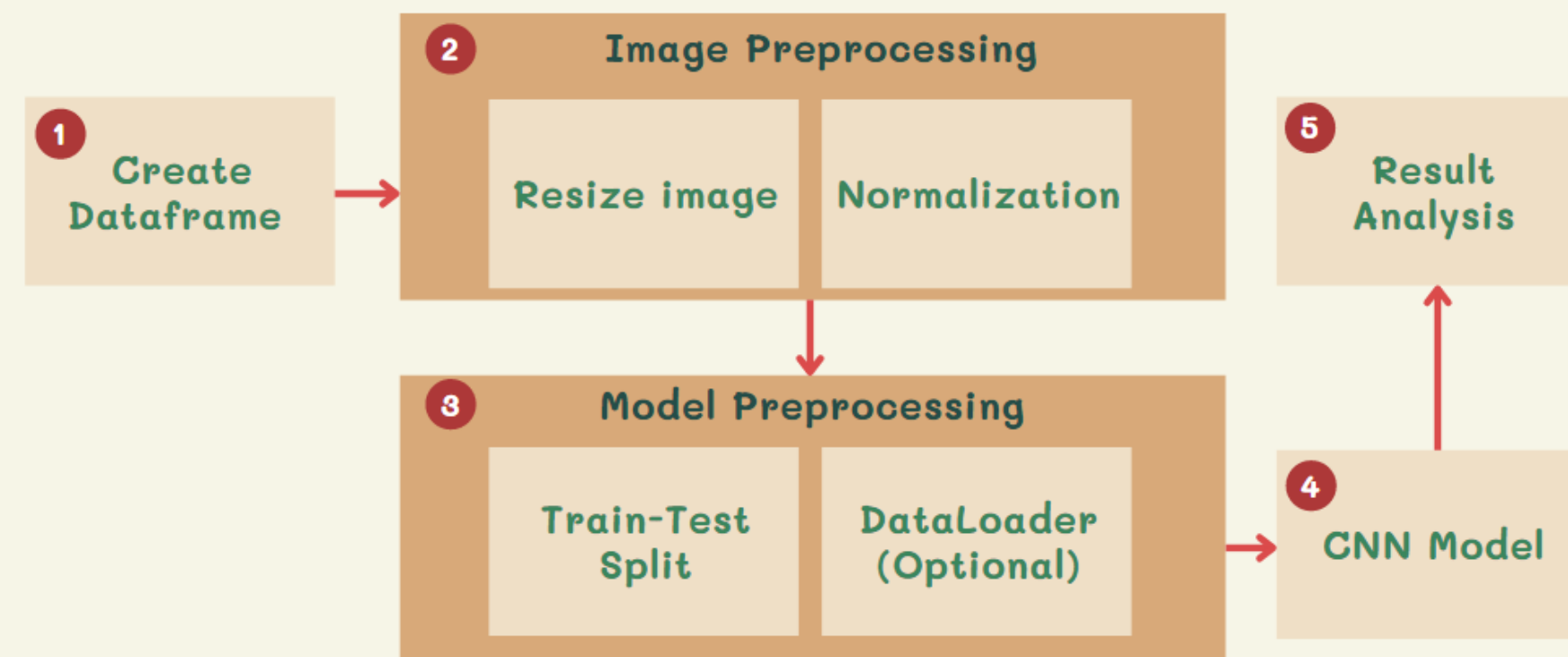
Training loss and Validation loss



## \* 4-5. CNN Model & Result analysis

### Model 3: Transfer Model (Resnet18) (Best Model)

- Change input channel to 1 (Grayscale)
- Change output channel



Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	$7 \times 7, 64, \text{stride } 2$
conv2_x	$56 \times 56 \times 64$	$3 \times 3 \text{ max pool, stride } 2$
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	$7 \times 7 \text{ average pool}$
fully connected	1000	$512 \times 1000 \text{ fully connections}$
softmax	1000	



## \* 4-5. CNN Model & Result analysis

### Training 1

#### Parameter:

optimizer = Adam

Learning rate = 0.0001

criterion = CrossEntropyLoss

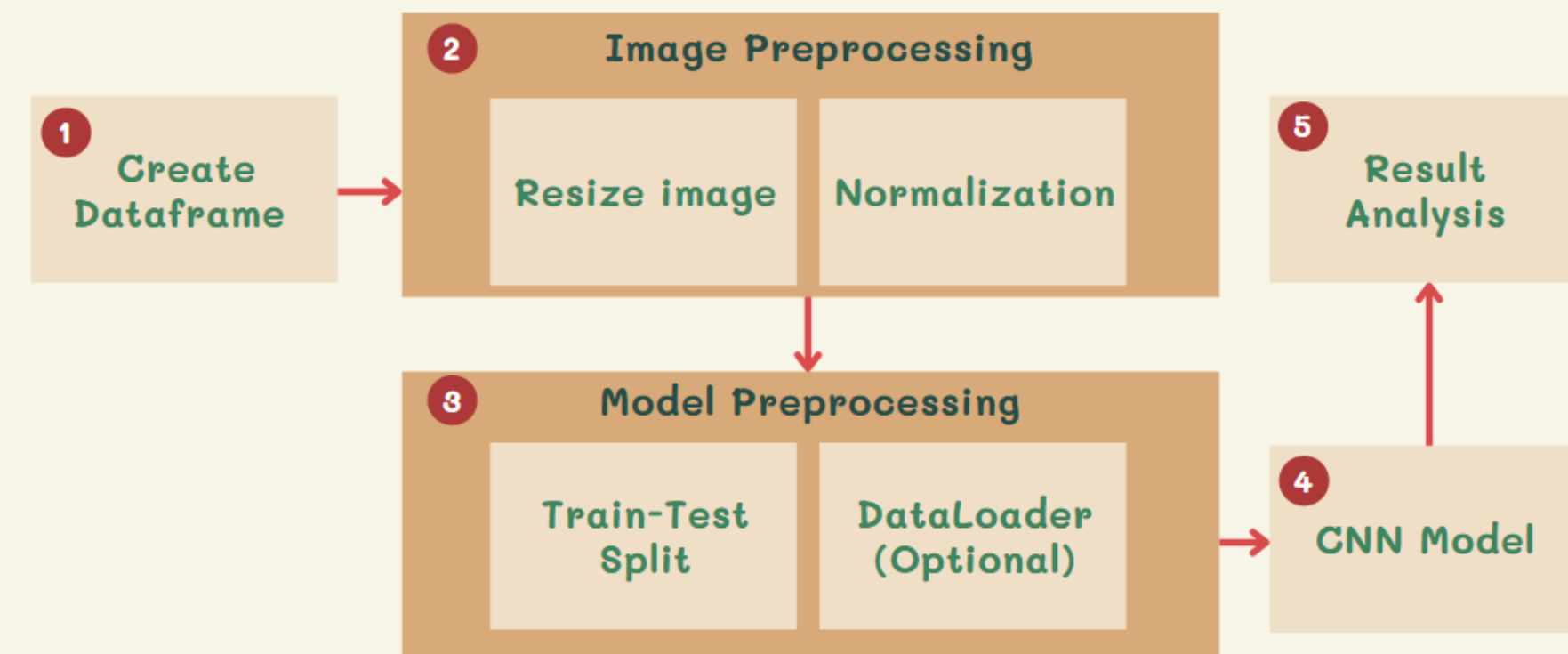
scheduler = ReduceLROnPlateau

number of epochs = 100

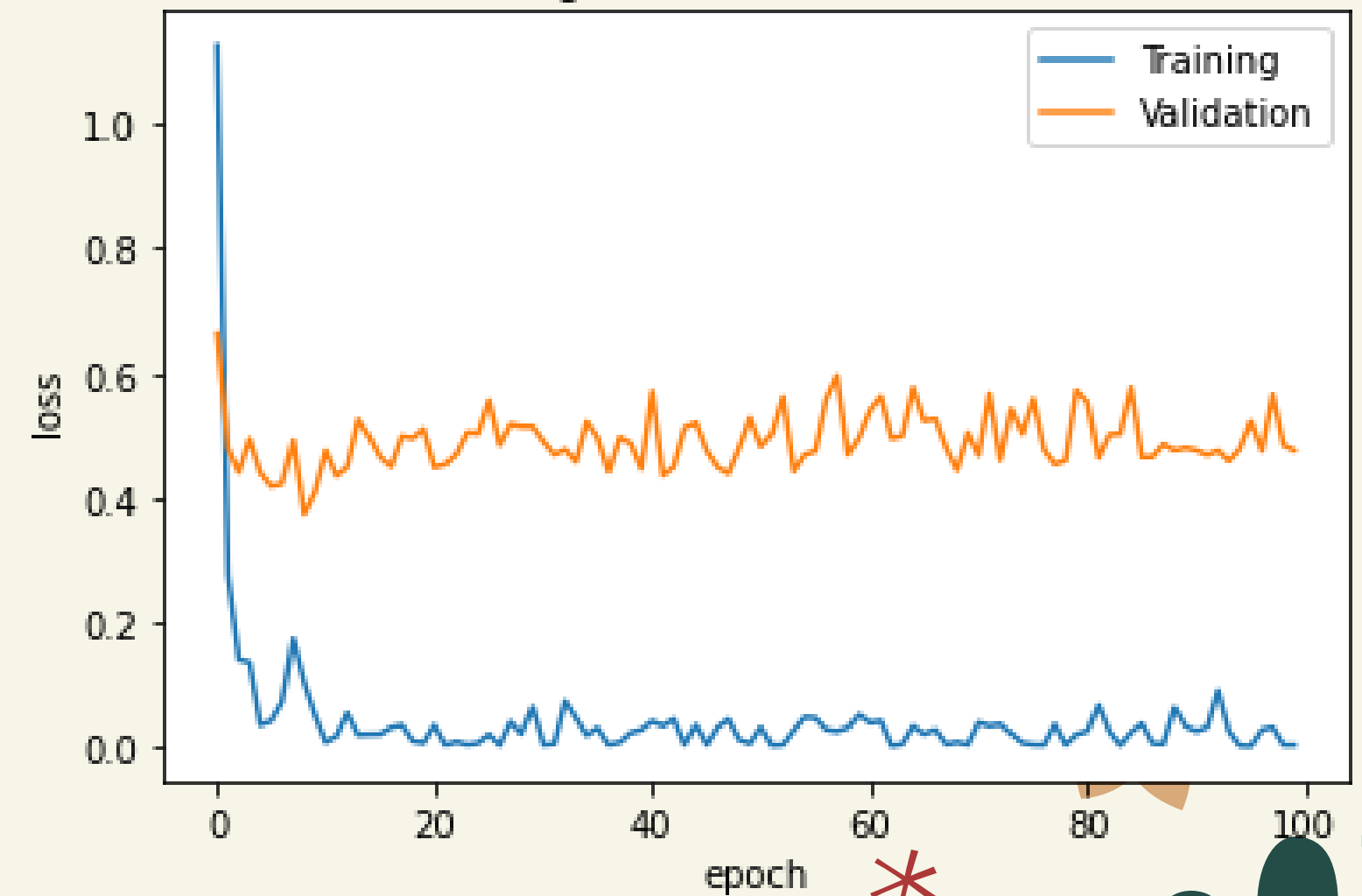
#### Accuracy:

Training accuracy: 100%

Validate accuracy: 89%



Training loss and Validation loss







# Model Improvement

1. Use image augmentation (crop/resize/change intensity level) to create more dataset
2. Use Grid search/Random search to find the best parameter for each model

# Reference

- <https://www.youtube.com/playlist?list=PL5-TkQAfAZFbzxjBHtzdVCWEOZbhomg7r>
- <https://www.kaggle.com/pranjalsoni17/natural-scene-classification>
- <https://www.analyticsvidhya.com/blog/2021/09/convolutional-neural-network-pytorch-implementation-on-cifar10-dataset/>
- [https://pytorch.org/tutorials/beginner/basics/buildmodel\\_tutorial.html](https://pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html)
- <https://www.kaggle.com/raddar/tuberculosis-chest-xrays-shenzhen>
- <https://www.pluralsight.com/guides/introduction-to-resnet>



THANK YOU