

## A stylized illustration of a person with dark skin and short dark hair, wearing a light purple hoodie and dark pants, sitting at a wooden desk. They are smiling and looking at a large black computer monitor. On the desk, there is a laptop with a red cover and a black screen, and a small white mouse. The background is white with several floating icons: a red head with a brain and gears, a speech bubble with three dots, a document with red and black lines, a play button, a microphone, and a document with a red outline. The desk has a simple wooden frame with four legs.

# A Project Presentation

## CMPE 258- Deep Learning

# TEAM MEMBERS



Neel Desai

017436575



Sai Praneeth  
Konuri

017425876



Mansi  
Vekariya

016996291



Samarth  
Sharma

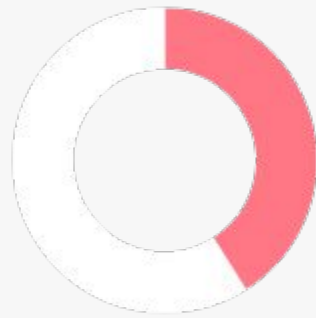
017103437

# Overview

- Build a CNN model capable to detecting if a computer chips has any defects or no
- Building a CNN which has at least 90% accuracy.
- Include MLOps in the training and Inference , we will be focusing: Experiment tracking, model registry, model versioning and model deployment and POC for retraining pipeline
- Link to dataset:  
<https://www.kaggle.com/datasets/qingyi/wm811k-wafer-map>

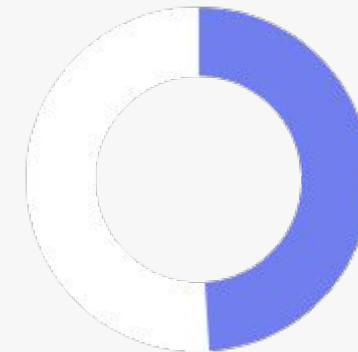


# Get to know the data



The data which we are using has the following classes for failures:

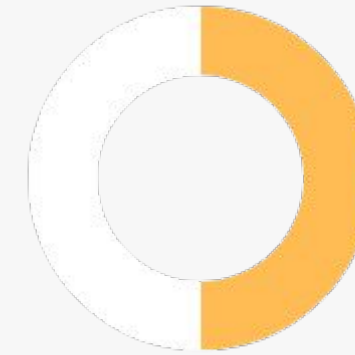
Center, Donut, Edge-Loc, Edge-Ring, Loc, Random, Scratch, Near-full, none



We have a total of 811457 wafer maps, spread across 632 image size

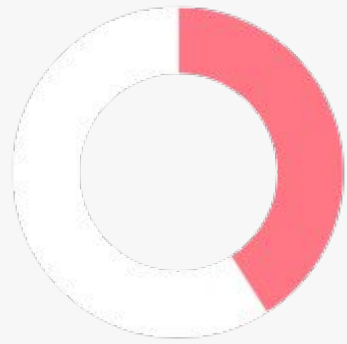


To make this consistent for the purpose of the project we will be looking at images with shape 26x26, we have 30078 such images

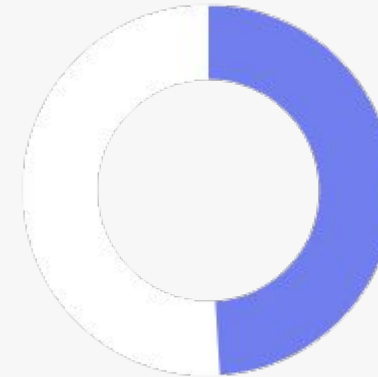


On exploring the distribution of labels across our images we found that we had major class imbalance, where the no error class account for more than 95% of the data and rest 8 classes of different defects accounted for remaining 5

## Get to know the data ( Continued )



To address the imbalance we pivoted from multi-class classification of different defects, to a binary classification problem i.e. identify if an image represents a chip with defect or no defect



Additional we performed downsampling to make sure that both classes are evenly distributed. We ended with having 900 images per each class

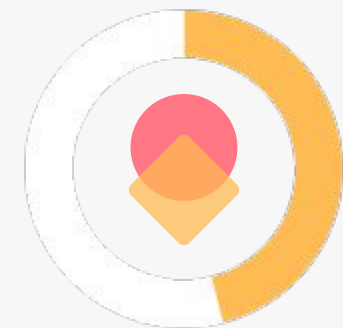
Note: we tried Image augmentation but the results were not that great, and we planned on running AutoTraining loop on Low cost cloud resources, so we decided to keep the train computation as light as possible.



# Training ( Overview )



We used ChatGPT to provide us a good starting point for a CNN architecture from where we can could built forward



First we perform training on colab to find architecture that work well



Then the best performing architectures will be used in Autotraining loop on Azure ML



Additionally we tracked all the experiments using MLFlow (even on colab) so that later we can easily register it in Azure Model registry



# Training: Starting Point

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(26, 26, 1)),
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu', padding='same'),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='Adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Starting Architecture

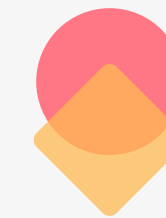
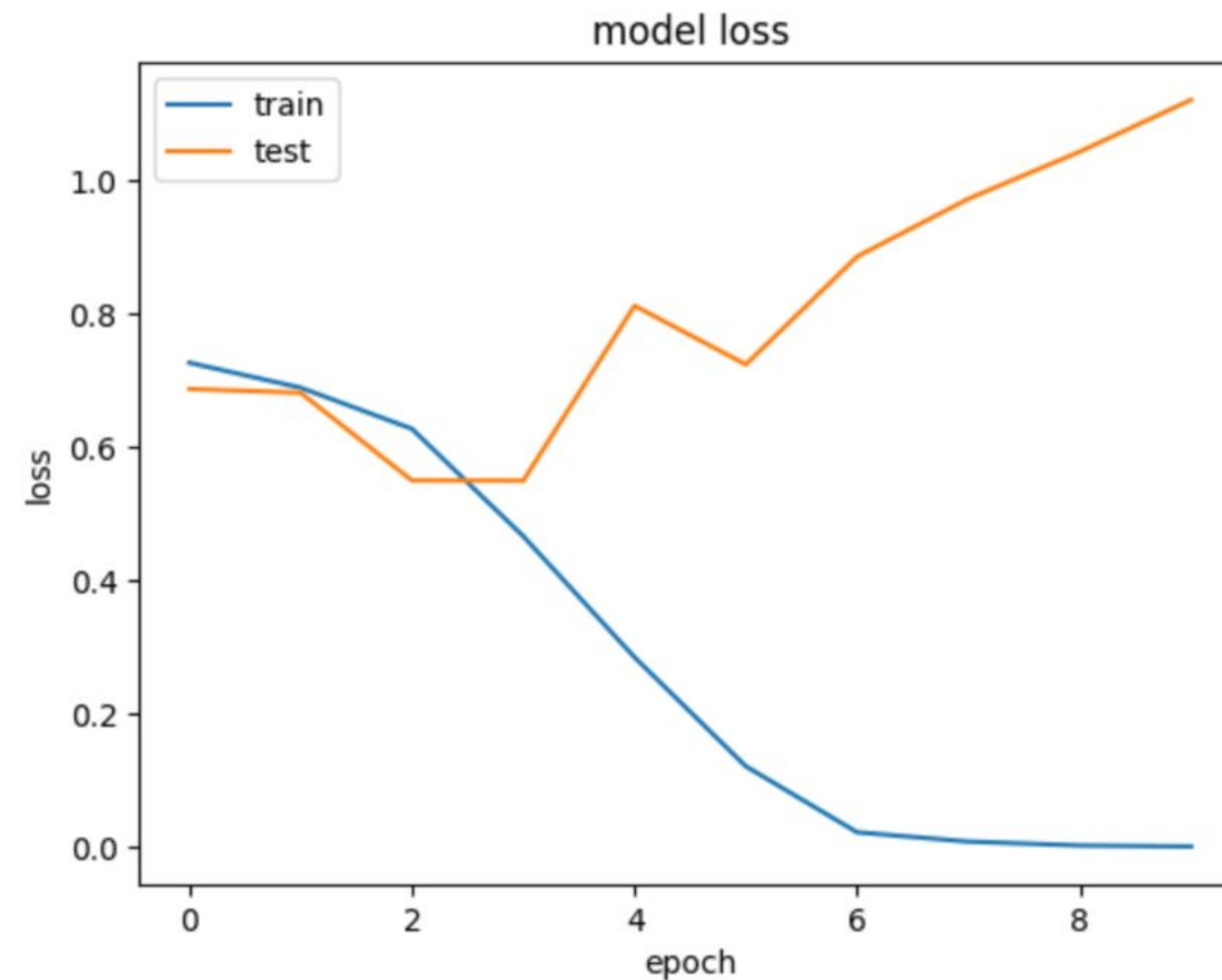


The network is  
overfitting on train set



It is not able to capture  
all the variability of the  
data

## Loss Curve



Val\_accuracy = 75%



Lets us try a couple of  
different things to fix it



# Further experimentation

For Model 2:

We have added pooling layers and a dropout layer to our model 1.

For Model 3:

We have changed the optimizer from Adam to Adadelata.

# Training: Ending Point, Best Model

```
[ ] model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(26, 26, 1)),
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu', padding='same'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer=tf.keras.optimizers.AdamW(), loss='binary_crossentropy', metrics=['accuracy'])
```

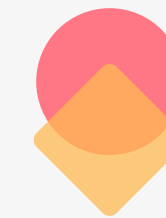
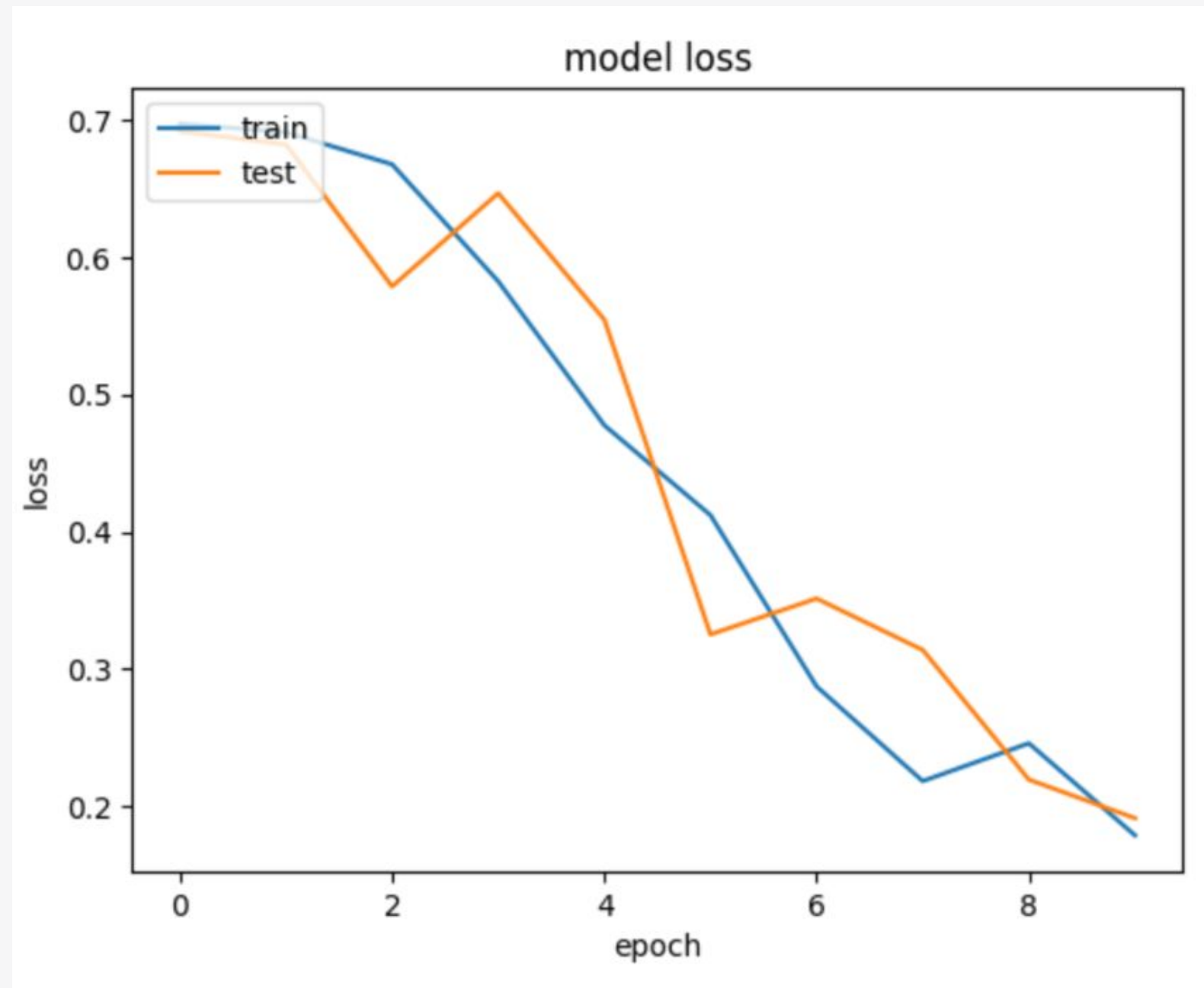


There is no more  
overfitting



Loss is continuing to go  
down, longer training  
could result in even  
better result

## Loss Curve



Val\_accuracy = 93%

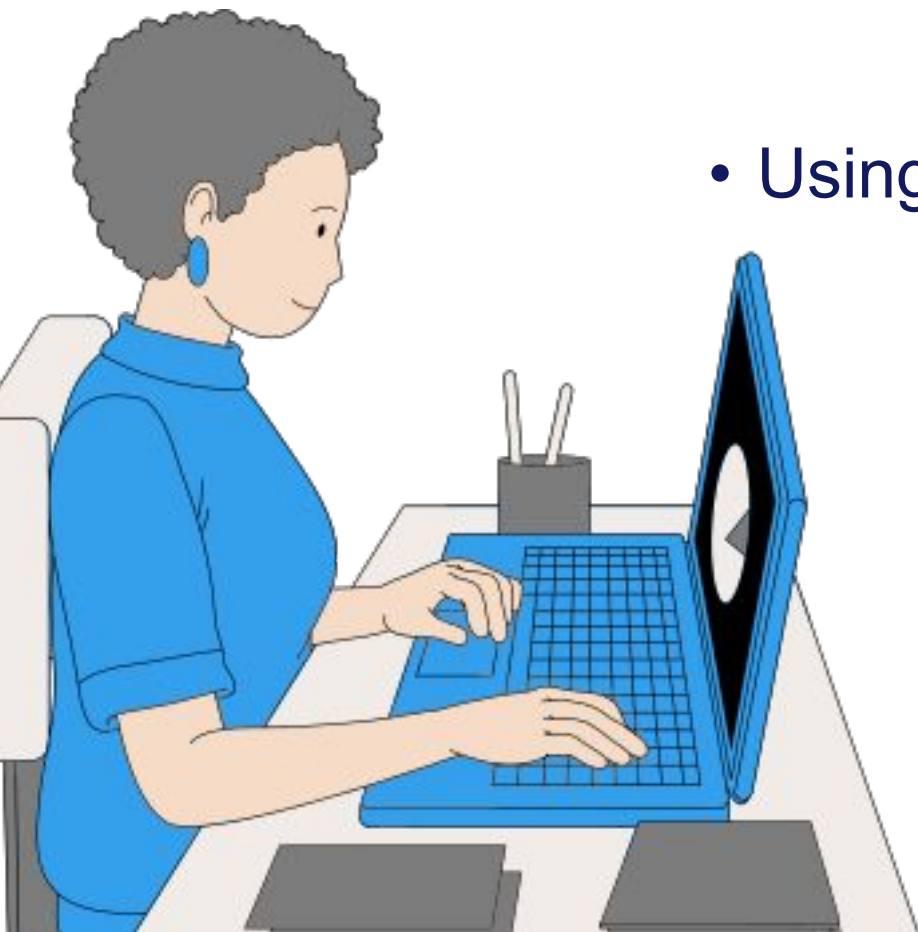


This model is significantly  
the better than our  
baseline. And this is  
where we stop the dev

# How did we get there

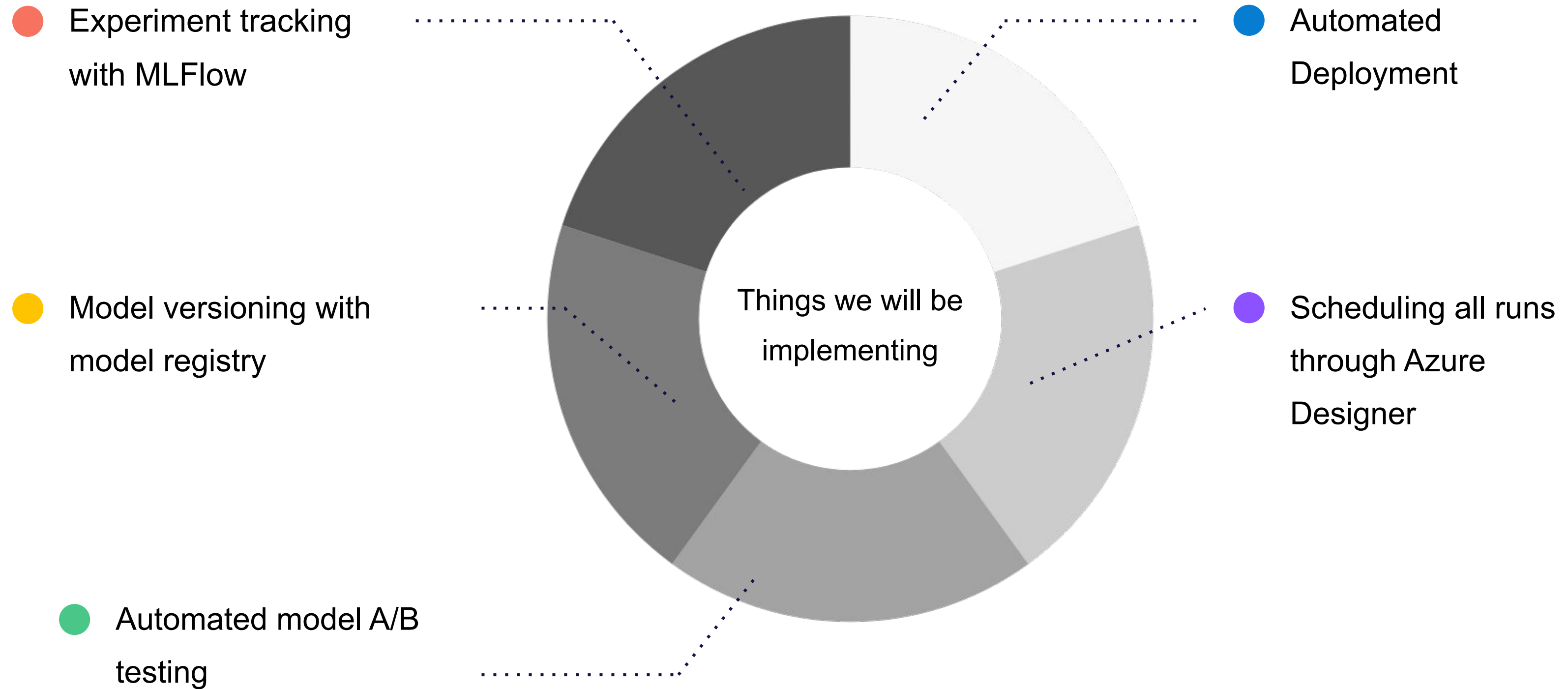
What changes did we make to get from 75% to 93% ?

- Using different regularization techniques such as Batch Normalization and Dropout layer
- Reducing the model complexity - specially after flattening the input
- Using newer more adaptive optimizer AdamW

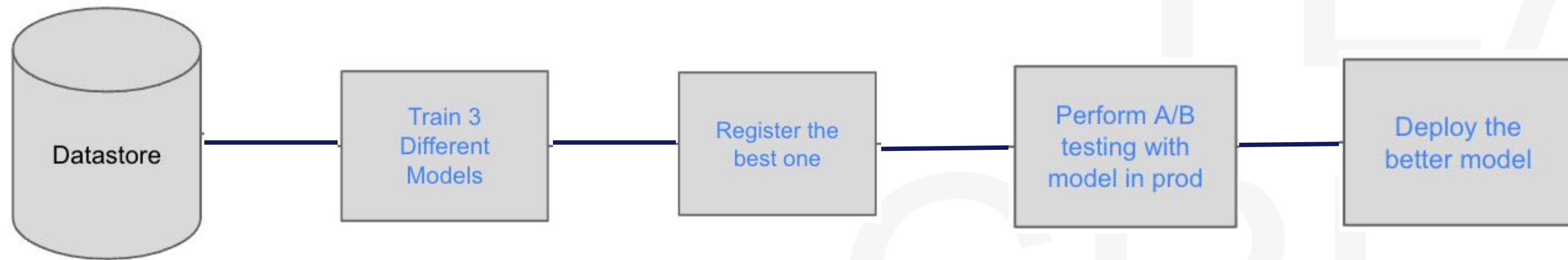




# ML Ops on Azure



# Training/Retraining Pipeline



- This pipeline focuses in auto training, testing and deployment, cause of which we assume that the data coming in the data store in clean data and doesn't require much data preprocessing.
- This pipeline can be scheduled to run on regular intervals , or run when new data arrives.
- It can configure directly as a pipeline on Azure ML.

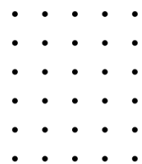
# Experiment Tracking

🔍 Search

🔗 Columns

Experiment	☆ Latest job	Last submitted ↓	Created
<a href="#">automated_training_loop</a>	<a href="#">258Project</a>	May 19, 2024 5:44 PM	May 19, 2024 5
<a href="#">wafer_defect_2024-05-20 00:08:13.326173</a>	<a href="#">model3</a>	May 19, 2024 5:10 PM	May 19, 2024 5
<a href="#">Automated_Pipeline</a>	<a href="#">258Project</a>	May 19, 2024 3:12 PM	May 19, 2024 3
<a href="#">iris_experiment</a>	<a href="#">ivory_ring_lwrwybn0</a>	May 19, 2024 2:34 PM	May 16, 2024 1
<a href="#">wafer_defect_2024-05-19 08:37:14.301825</a>	<a href="#">model3</a>	May 19, 2024 1:39 AM	May 19, 2024 1
<a href="#">iris_experiment_2024-05-18 00:34:41.819518</a>	<a href="#">model3</a>	May 17, 2024 5:41 PM	May 17, 2024 5
<a href="#">iris_experiment_2024-05-17 18:56:50.860243</a>	<a href="#">model2</a>	May 17, 2024 11:57 AM	May 17, 2024 1

An experiment is nothing but a collection of different models you built when trying to find the best one



# MLFlow

waffer\_defect\_2024-05-20 00:08:13.326173

+ Create job

↺ Refresh

⬇ Export

✕ Cancel




🗑 Delete

⚙ View options

⋮

🔍 Search

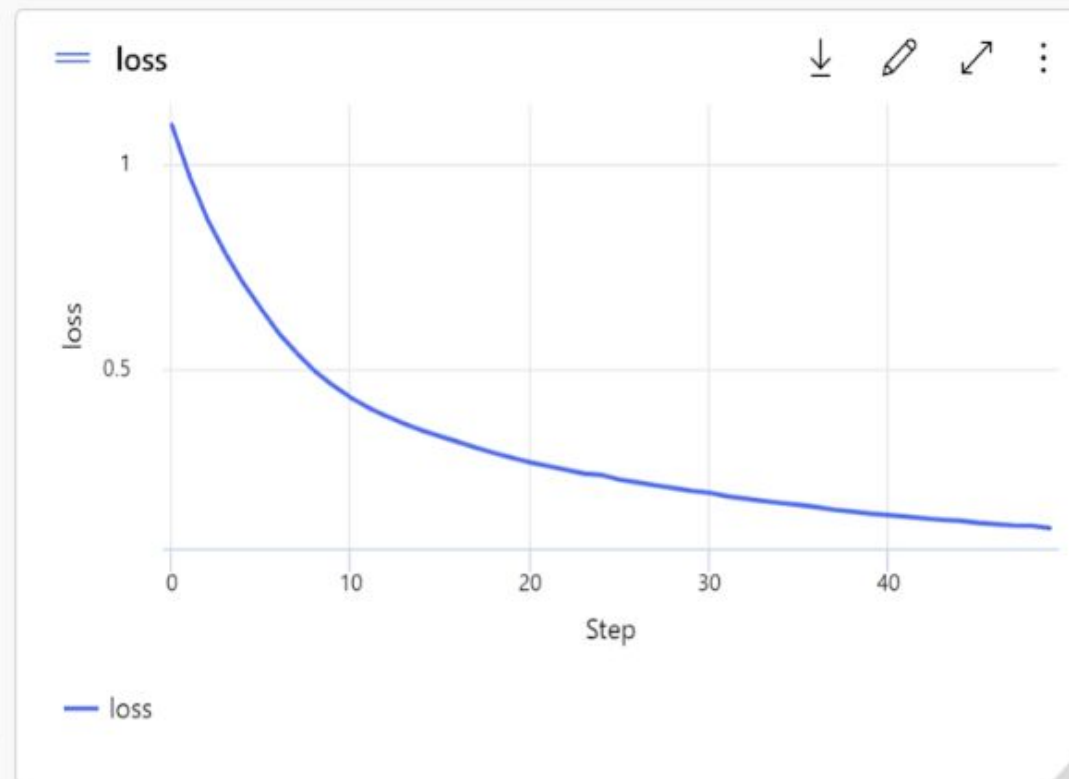
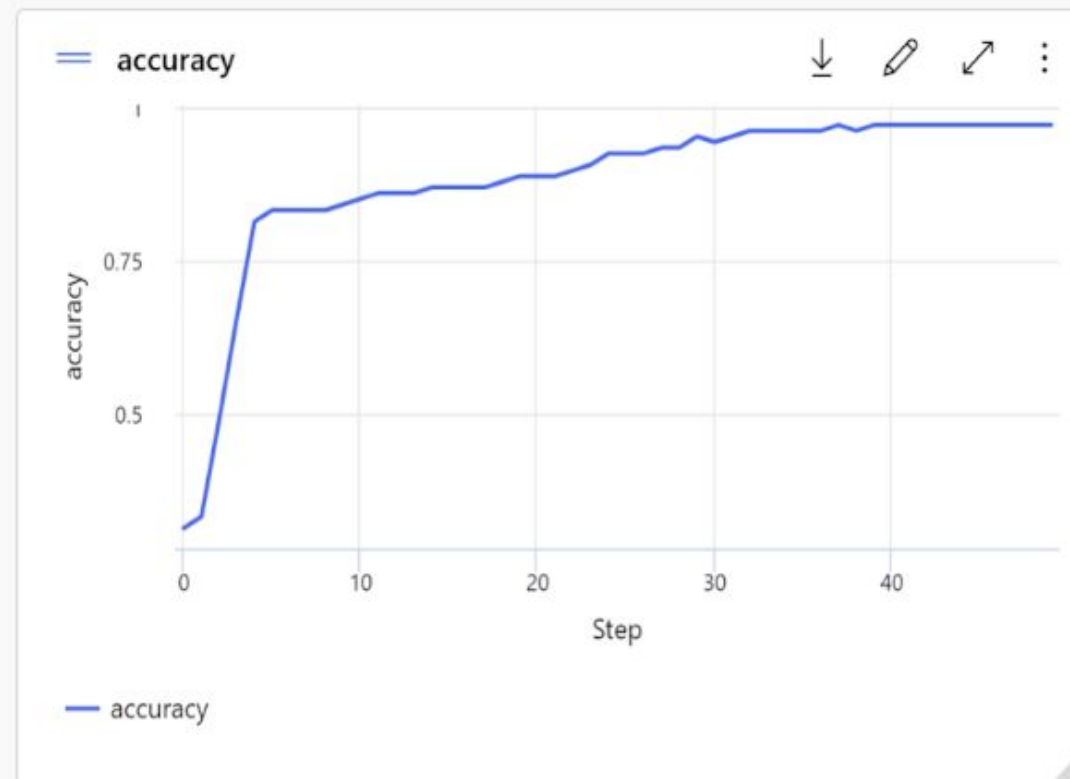
🔍

👁 Display name (3 visualized)	Parent job name	Status	Created on ↓
👁  model3		✅ Completed	May 19, 2024 5:10 PM
👁  model2		✅ Completed	May 19, 2024 5:09 PM
👁  model1		✅ Completed	May 19, 2024 5:08 PM

With each experiment , we have the list of all the models trained for it, we use MLFlow for logging models performance and its artifacts



# Artifacts and metrics



For each logged model we have all the logged artifacts and metrics, along with the hyperparameter used for training it

# Model Registry

Default Directory > Team-pi > Models

Model List

+ Register ▾

↻ Refresh

🗑 Delete

📁 Archive

▶ Deploy

⚖ Compare (preview) ▾

⋮

☒ Show latest versions only

☐ Include archived

🔍 Search

≡ Filter

📄 Columns

Name	☆	Version	Type	Source	Experiment	Job (Run ID)
<a href="#">wafer-defect</a>		3	MLFLOW	This workspace	wafer_defect_2024-05-20 00:0...	<a href="#">14b40f5a-c291-4</a>
<a href="#">iris_model</a>		6	MLFLOW	This workspace	iris_experiment	<a href="#">82fa037c-e081-4</a>
<input type="radio"/> <a href="#">iris_model</a>	☆	5	MLFLOW	This workspace	iris_experiment	<a href="#">78059ec5-8d8b-</a>
<a href="#">wafer-defect</a>		2	MLFLOW	This workspace	wafer_defect_2024-05-19 08:3...	<a href="#">d4ffa8e8-8cdf-4e</a>
<a href="#">wafer-defect</a>		1	MLFLOW	This workspace		
<a href="#">iris_model</a>		4	MLFLOW	This workspace	iris_experiment_2024-05-18 00...	<a href="#">ff7d009b-6e74-4</a>
<a href="#">iris_model</a>		3	MLFLOW	This workspace	iris_experiment_2024-05-17 18...	<a href="#">1b355be1-23f6-4</a>

It's a means through which we can apply a version control on our models, it could be as simple as same model trained on data at 2 different times

# Endpoint Deployment

Default Directory > Team-pi > Endpoints > team-pi-vtzdu

team-pi-vtzdu

Details

Test

Consume

Monitoring

PREVIEW

Logs

+ Add deployment

↻ Refresh

✎ Update traffic

🗑 Delete

Endpoint attributes

Service ID

team-pi-vtzdu

Description

--

Provisioning state

Succeeded

Error details

--

Compute type

Managed

Created by

Neel Desai

Created on

May 17, 2024 9:51 AM

Last updated on

May 17, 2024 9:51 AM

Authentication type

Key

Deployment summary

Live traffic allocation

✔ waffer-defect-1 (100%)

Mirrored traffic allocation

--

Deployment waffer-defect-1

Name

waffer-defect-1

Live traffic

100%

Scoring script

Auto-generated

Provisioning state

✔ Succeeded

Error details

--

SKU

Allows us to expose  
our model as an API to  
the rest of the world.

# Inference Pipeline





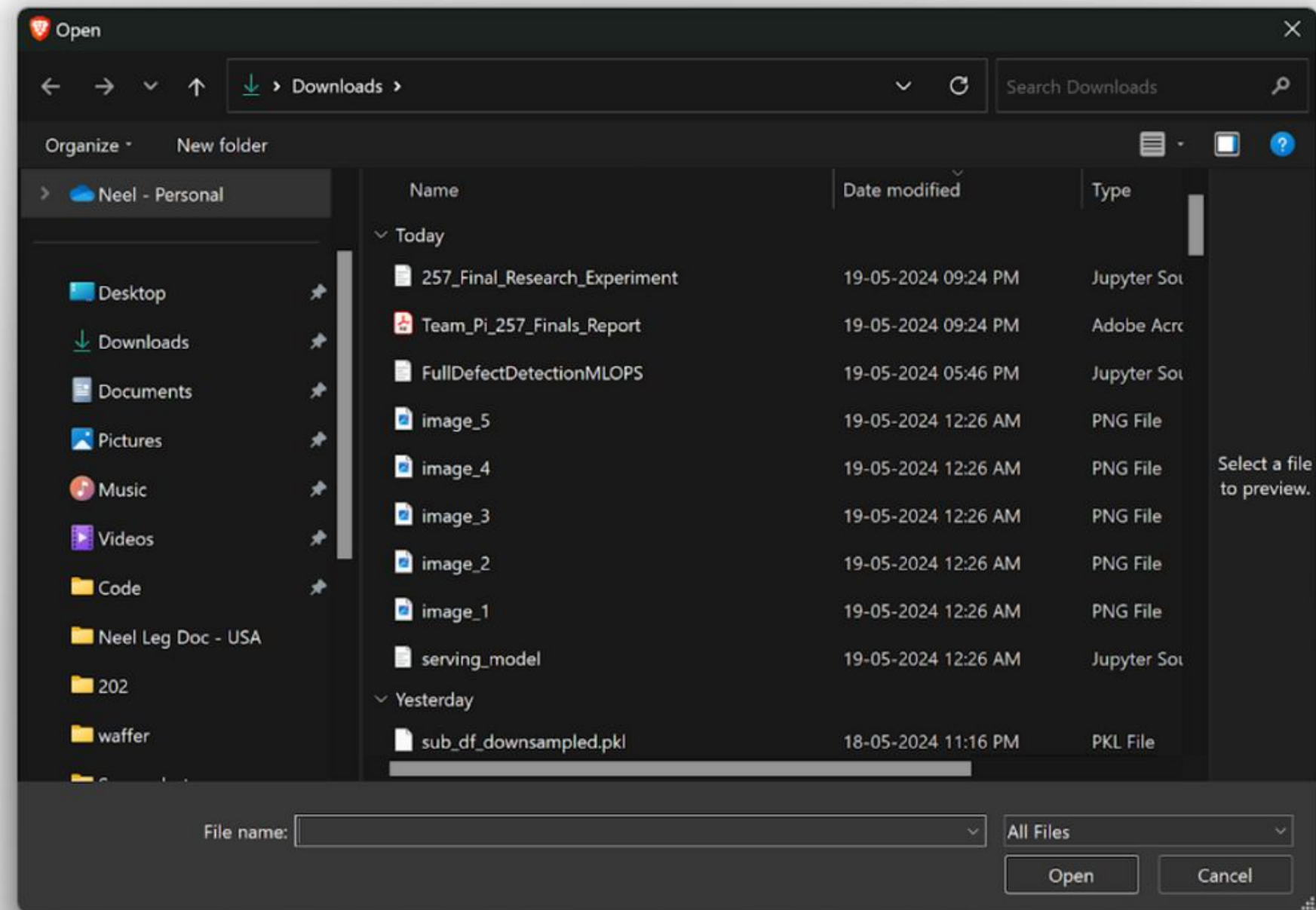
# WebPage Mockup

## Upload an Image for Prediction

Choose File

No file chosen

Upload



# WebPage Interface

## Upload an Image for Prediction

No file chosen

**Predicted Class: Normal**

# WebPage Interface

## Upload an Image for Prediction

No file chosen

**Predicted Class: Defect Detected**

The background features a series of overlapping circles in red, blue, orange, and teal. Vertical lines of varying heights and colors (white, light blue, light green) are positioned behind the circles. On the left, there are several concentric, thin-lined arches. On the right, there is a small grid of dots.

**Thank You!!**