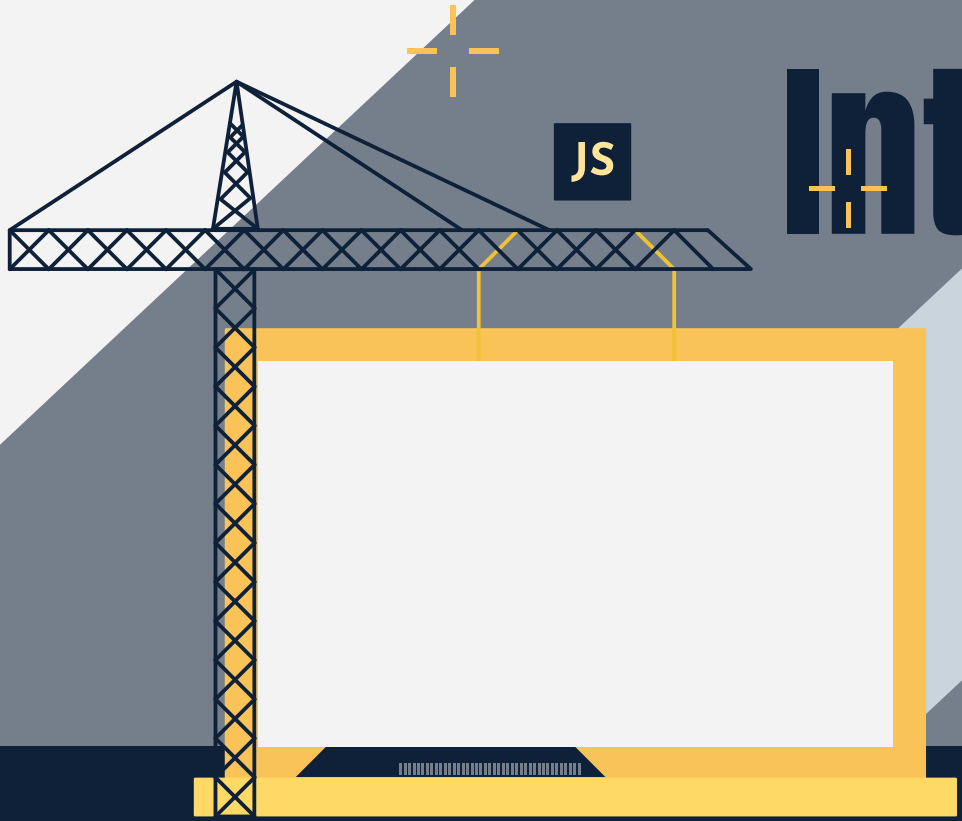


IntelliOre S1

Mina Sung

Hyoyeon Lee

TABLE OF CONTENTS

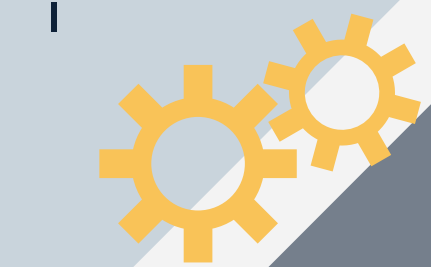
Problem

“Over 40% of the most serious injuries in mining involve incidents classified as caught-in machinery.”

Market Analysis

1. **Tech adoption** is becoming more prevalent in mining industry. Such AI, automation, and drones are being incorporated into the process.
2. The mining industry is **experiencing steady growth**, projected to hit $2.5-$3.7 trillion by 2029-2033 with ~5-6% compound annual growth rate.
3. The average age in the mining industry tends to be higher than the general workforce, typically falling between 40 and 50, those who are **more susceptible to injuries**.

Main Competitors



HEXAGON

Honeywell

blacklinesafety

- Advanced AI and safety systems, but expensive and difficult to use.
- IntelliOre simplifies predictive safety into clear, actionable alerts instead of complex dashboards.

- Hardware focused with Accurate sensors, but outdated UX and complex data.
- IntelliOre turns sensor data into user-centered risk insights, prioritizing software and usability.

- Strong real-time alerts with monitoring, but limited early prediction.
- IntelliOre focuses on predicting risks before incidents happen, not just reacting.

OUR SOLUTIONS

MVP:

Minimum version of product

Core Features of IntelliOre MVP

- Classification and detection of the image
- Analysis of the mining machine data
- Visualization of mining machine data

Human Centered Design

- Users: Workers at the mining site who need to quickly and clearly understand machine breakdown prediction statuses.
- Aim: To build a website with clear information organization and effective visual elements.

OUR SOLUTIONS

Product Iteration

Season 0:

- Safety of workers
- Machine breakdown prediction

Season 1:

- Technical aspect: ANN, Data visualization technique, CNN, website
- Decision making process about core features
- E.g) safety gear image detection, visualization of the machine conditions, alerts for machine breakdown

OUR Logo

Color BLUE

Trust

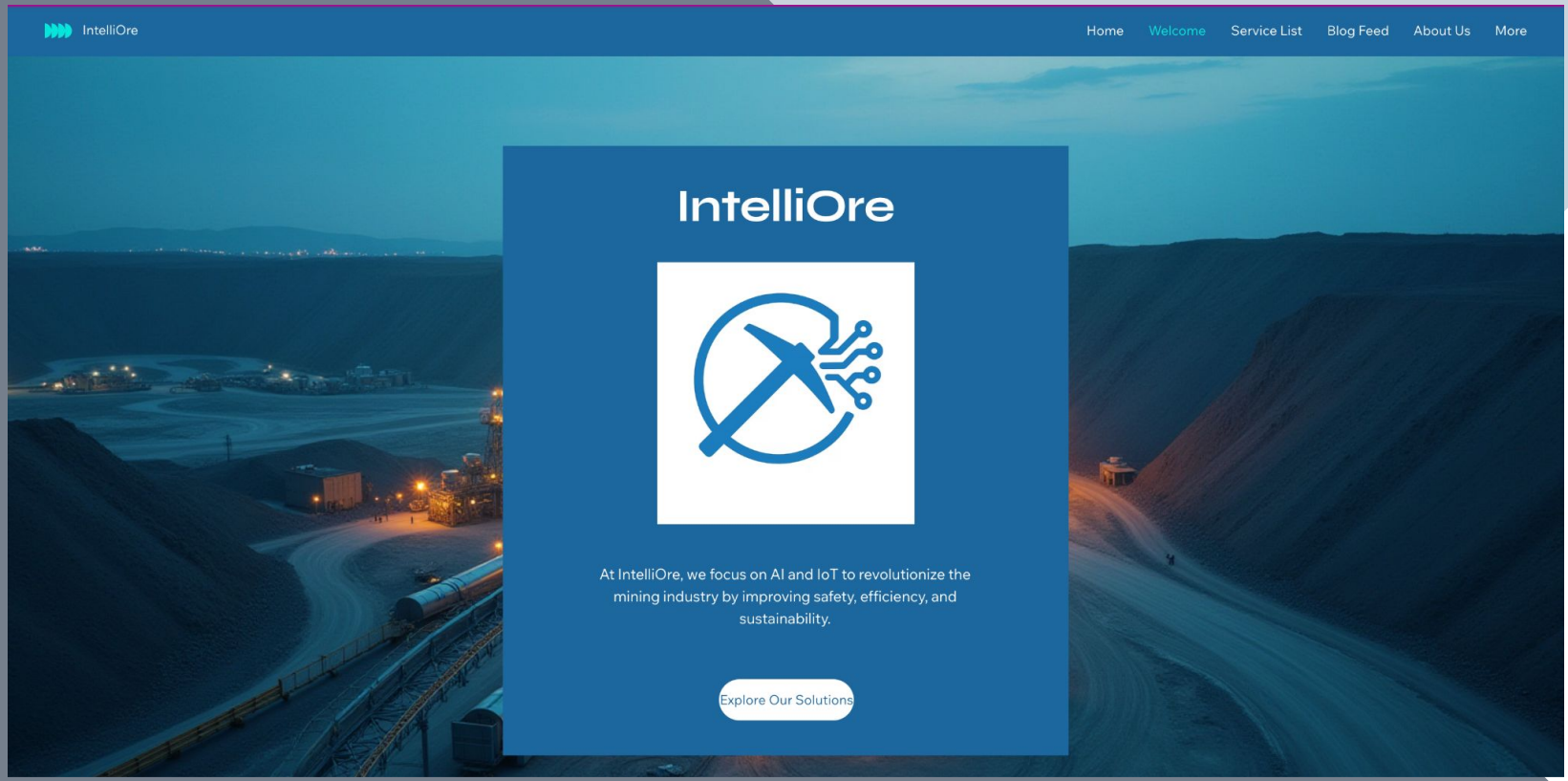
Protection

Clarity

What our logo communicates

- Industrial industry (mining industry)
- Data analysis / Information technology (IT)

Website


IntelliOre
Home
Welcome
Service List
Blog Feed
About Us
More
IntelliOre
At IntelliOre, we focus on AI and IoT to revolutionize the mining industry by improving safety, efficiency, and sustainability.
Explore Our Solutions

What's Next

Tech

Build the app prototype

Business

- Translate problem–solution logic into a coherent product narrative.
- Strengthen positioning and differentiation within the mining safety ecosystem.
- Support MVP definition with clearer user journeys and use cases.


Moving onto the tech...

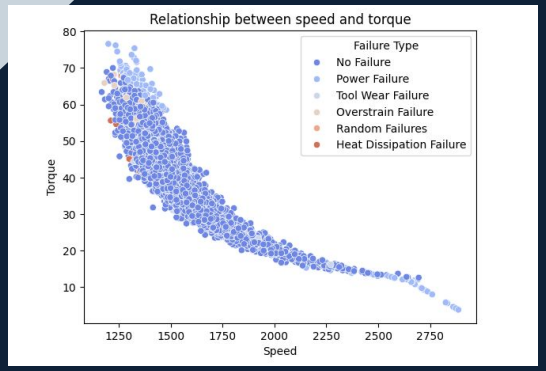
Tech Presentation

Predictive maintenance data - Mina

- Product ID Type
- Air temperature
- Process temperature
- Rotational speed
- Torque
- Tool wear
- Target Failure Type
- Air temperature
- Process temperature

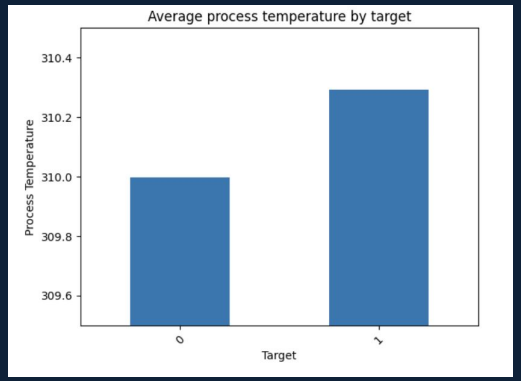
```
df= pd.read_csv('predictive_maintenance.csv')
print(df.head())
print(df.describe())
```

```
   UDI Product ID Type  Air temperature [K]  Process temperature [K]  \
0    1         M14860  M                298.1                    308.6
1    2         L47181  L                298.2                    308.7
2    3         L47182  L                298.1                    308.5
3    4         L47183  L                298.2                    308.6
4    5         L47184  L                298.2                    308.7

   Rotational speed [rpm]  Torque [Nm]  Tool wear [min]  Target Failure Type
0                    1551         42.8                0       0   No Failure
1                    1408         46.3                3       0   No Failure
2                    1498         49.4                5       0   No Failure
3                    1433         39.5                7       0   No Failure
4                    1408         40.0                9       0   No Failure
                UDI  Air temperature [K]  Process temperature [K]  \
count  10000.00000          10000.000000             10000.000000
mean    5000.50000            300.004930               310.005560
std     2886.89568              2.000259                 1.483734
min        1.00000            295.300000               305.700000
25%     2500.75000            298.300000               308.800000
50%     5000.50000            300.100000               310.100000
75%     7500.25000            301.500000               311.100000
max    10000.00000            304.500000               313.800000

       Rotational speed [rpm]  Torque [Nm]  Tool wear [min]        Target
count            10000.000000  10000.000000     10000.000000  10000.000000
mean              1538.776100     39.986910       107.951000      0.033900
std                179.284096      9.968934        63.654147      0.180981
min               1168.000000      3.800000         0.000000      0.000000
25%               1423.000000     33.200000        53.000000      0.000000
50%               1503.000000     40.100000       108.000000      0.000000
75%               1612.000000     46.800000       162.000000      0.000000
max               2886.000000     76.600000       253.000000      1.000000
```



The plot graph that shows the relationship between speed and torque

- If the speed decreases, the torque decreases.



The bar graph that shows the average temperature for the different targets.

0 = Normal
1 = Failure

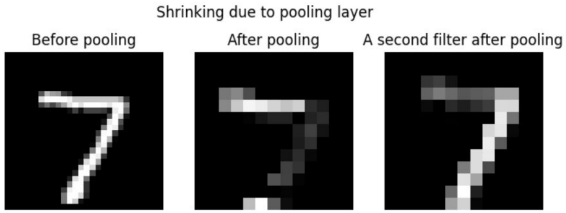
CNN (MNIST) - Mina

Simple CNN: Input -> Conv -> Flatten -> Dense -> Output

Complex CNN: Input -> Conv -> Pooling -> Conv -> Pooling -> Flatten -> Dense -> Output

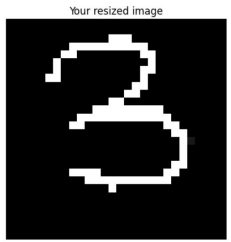
```
#pooling shrinks the image and extracts the important features
#Before and after pooling
from tensorflow.keras import Model
#a model that outputs the pooling layer output
pool_model = Model(inputs=model_cnn2.input, outputs=model_cnn2.layers[2].output) # the second layer is the fi
pooled = pool_model.predict(x_test_cnn[0:1])

print("Shape before pooling layer", (26,26,32))
print("Shape after pooling layer", pooled.shape[1:])

plt.figure(figsize=(8,3))
plt.subplot(1,3,1); plt.imshow(x_test_cnn[0, :, :, 0], cmap='gray'); plt.title("Before pooling"), plt.axis("o
plt.subplot(1,3,2); plt.imshow(pooled[0, :, :, 0], cmap='gray'); plt.title("After pooling"), plt.axis("off")
plt.subplot(1,3,3); plt.imshow(pooled[0, :, :, 1], cmap='gray'); plt.title("A second filter after pooling"), |
plt.suptitle("Shrinking due to pooling layer")
plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step
Shape before pooling layer (26, 26, 32)
Shape after pooling layer (13, 13, 32)
```



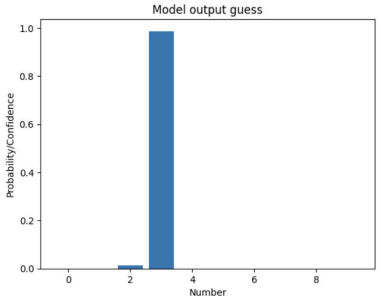
Pooling: reduces the dimensionality of the data

```
#Loading and preprocessing of the image : Reshape it into 28*28 and make it grey scale with black background

import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image

#Load the image
path="1.png"
img = image.load_img(path, target_size=(28,28), color_mode="grayscale")

#Convert the image
img_array = image.img_to_array(img) #Convert image into a numerical array
img_array = 255 - img_array # Inver the image colors to a black background and white lines
img_array = img_array/255.0 # Greyscale to Binary (0~255) -> (0~1)
img_array = img_array.reshape(1, 28, 28, 1) # reshaping to match the cnn input

#Show the processes data
plt.imshow(img_array[0, :, :, 0], cmap='gray')
plt.title("Your resized image")
plt.axis("off")
plt.show()
```



I used an image of my hand-drawn number.

```
#predict the number in the image
probs = model_cnn2.predict(img_array, verbose=0)[0]
pred = probs.argmax()

#Show a graph of the predictions and the model's confidence of each output possibility
plt.bar(range(10), probs)
plt.title("Model output guess")
plt.xlabel("Number")
plt.ylabel("Probability/Confidence")
plt.show()
```



The bar graph that shows the model's confidence of each output possibility.

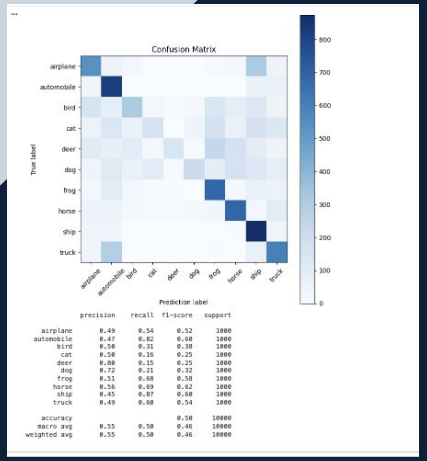
CNN (CIFAR) - Mina

MNIST and CIFAR are both used for image classification

MNIST
- **grayscale images of handwritten digits (0-9)**

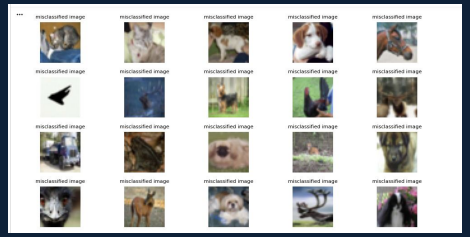
CIFAR
- **color images of real-world objects**

```
#See an image of each category
labels = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
plt.figure(figsize=(10,6))
for i in range(10):
    plt.subplot(2,5,i+1)
    plt.imshow(x_train[i])
    plt.title(labels[y_train[i][0]])
    plt.axis('off')
plt.tight_layout()
plt.show()
```

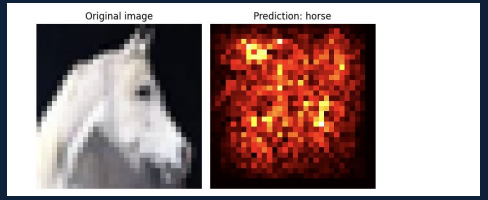


	precision	recall	f1-score	support
airplane	0.49	0.54	0.52	1000
automobile	0.47	0.82	0.60	1000
bird	0.50	0.31	0.38	1000
cat	0.50	0.16	0.25	1000
deer	0.80	0.15	0.25	1000
dog	0.72	0.21	0.32	1000
frog	0.51	0.68	0.58	1000
horse	0.56	0.69	0.62	1000
ship	0.45	0.87	0.60	1000
truck	0.49	0.60	0.54	1000
accuracy			0.50	10000
macro avg	0.55	0.50	0.46	10000
weighted avg	0.55	0.50	0.46	10000

- Dark blue cells on the diagonal mean the model predicted that class correctly many times.
- Some classes (automobile, ship, horse) are predicted well.
- Other classes (cat, deer, dog) are often misclassified.



Show the misclassified images



Discover the important pixel in the image that helps the model make the prediction

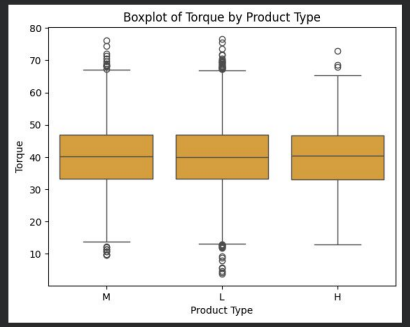
Tech - Hyoyeon

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

df = pd.read_csv ('predictive_maintenance.csv')

print("Shape of dataframe:", df.shape)

# 5 rows of the data
print(df.head(5))

# feature types
print(df.dtypes)
```

- Validating dataset
- Analyzes distributions of key sensor variables

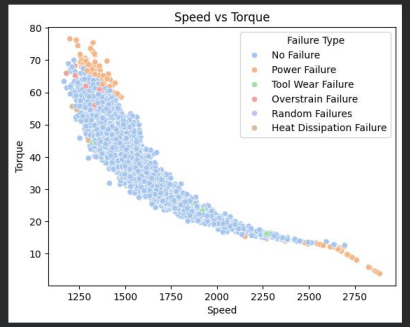
```
Shape of dataframe: (10000, 10)
   UDI Product ID Type  Air temperature [K]  Process temperature [K]  \
0    1     M14860    M                298.1                    308.6
1    2     L47181    L                298.2                    308.7
2    3     L47182    L                298.1                    308.5
3    4     L47183    L                298.2                    308.6
4    5     L47184    L                298.2                    308.7

   Rotational speed [rpm]  Torque [Nm]  Tool wear [min]  Target Failure Type
0                    1551         42.8                0       0   No Failure
1                    1408         46.3                3       0   No Failure
2                    1498         49.4                5       0   No Failure
3                    1433         39.5                7       0   No Failure
4                    1408         40.0                9       0   No Failure
UDI                        int64
Product ID                object
Type                      object
Air temperature [K]      float64
Process temperature [K]  float64
Rotational speed [rpm]     int64
Torque [Nm]              float64
Tool wear [min]            int64
Target                     int64
Failure Type              object
dtype: object
```

```python
#Box plot of Torque by Product Type
sns.boxplot(x="Type", y="Torque [Nm]", data=df, color="orange")
plt.title("Boxplot of Torque by Product Type")
plt.xlabel("Product Type")
plt.ylabel("Torque")
plt.show()
```



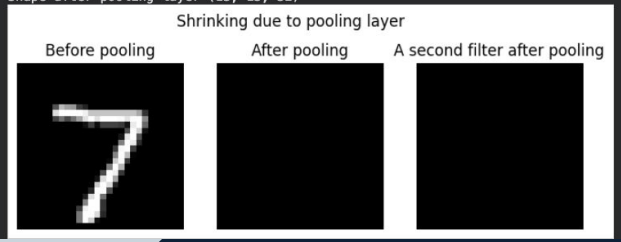
Exploratory analysis of different variables

```python
#Scatter plot of Speed vs Torque colored by Failure
sns.scatterplot(x=df["Rotational speed [rpm]"],y=df["Torque [Nm]"], hue=df["Failure Type"],palette="pastel")

plt.title("Speed vs Torque")
plt.xlabel("Speed")
plt.ylabel("Torque")
plt.show()
```



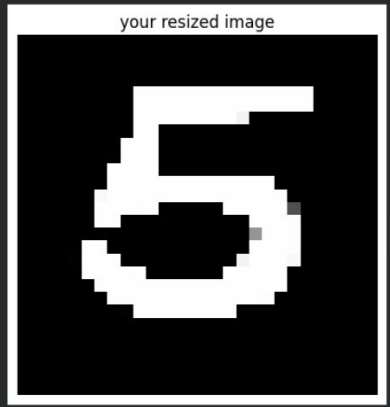
```python
#Before and after pooling
from tensorflow.keras import Model

#a model that outputs the pooling layer out
pool_model = Model(inputs=model_cnn2.input,
                   outputs=model_cnn2.layers[2].output) #the second layer is the f

pooled = pool_model.predict(x_test_cnn[0:1])

print("Shape before pooling layer", [26,26,32])
print("Shape after pooling layer", pooled.shape[1:])

plt.figure(figsize=(8,3))
plt.subplot(1,3,1)
plt.imshow(x_test_cnn[0,:,:,0], cmap="gray")
plt.title("Before pooling")
plt.axis('off')

plt.subplot(1,3,2); plt.imshow(pooled[0,:,:,0], cmap="gray")
plt.title("After pooling")
plt.axis('off')

plt.subplot(1,3,3); plt.imshow(pooled[0,:,:,1], cmap="gray")
plt.title("A second filter after pooling")
plt.axis('off')

plt.suptitle("Shrinking due to pooling layer")
plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
Shape before pooling layer [26, 26, 32]
Shape after pooling layer (13, 13, 32)
```



Visualizes how a pooling layer reduces image size while preserving key features

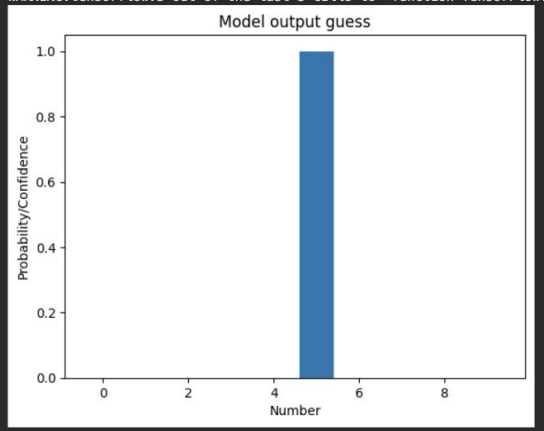
```python
#Loading and preprocessing of the image: Reshape it into 28*28 and make it greyscale with black background

import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image

#Load the image
path='5.jpg'
img = image.load_img(path, color_mode='grayscale', target_size=(28,28))

#Convert the image
img_array = image.img_to_array(img) #Convert image into a numerical array
img_array = 255 - img_array #Invert the image colors to a black BG and white lines
img_array = img_array/255.0 #Greyscale to Binary (0~255)-->(0~1)
img_array = img_array.reshape(1,28,28,1) #Reshaping to match the can input

#Show the processes data
plt.imshow(img_array[0,:,:,0], cmap="gray")
plt.title("your resized image")
plt.axis('off')
plt.show()
```



- Converts a handwritten digit into the 28x28 grayscale format
- uses the CNN to predict the digit from the processed image

```python
#predict the number in the image
probs = model_cnn2.predict(img_array, verbose=0) [0]
pred = probs.argmax()

#Show a graph of the predictions and the model's confidence of each output possibility

plt.bar(range(10),probs)
plt.title("Model output guess")
plt.xlabel("Number")
plt.ylabel("Probability/Confidence")
plt.show()
```

```
WARNING:tensorflow:5 out of the last 5 calls to <function TensorFlowTrainer.make_predict_
```

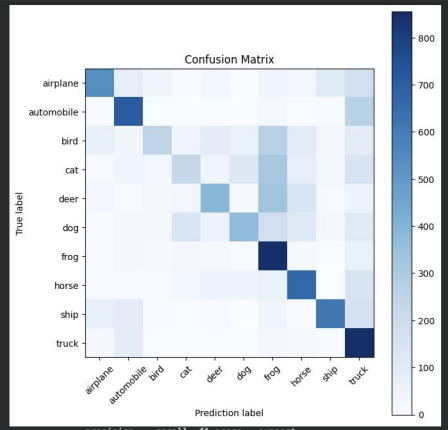


- Displays the model's confidence for each digit (0–9).
- Highlights which prediction the model is most confident in.

```python
#See an image of each category
labels = ["airplane","automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

plt.figure(figsize=(10, 10))
for i in range(10):
    plt.subplot(2,5, i+1)
    plt.imshow(x_train[i])
    plt.title(labels[y_train[i][0]])
    plt.axis('off')
plt.tight_layout()
plt.show()
```

- Confirms what each CIFAR-10 category looks like in the dataset.
- Ensures the model is trained on visually distinct object classes.

```python
#confusion matrix
from sklearn.metrics import confusion_matrix, classification_report
import numpy as np
import matplotlib.pyplot as plt

probs = model.predict(x_test, verbose= 0)
pred = np.argmax(probs, axis=1) # convert the probs into a 10 classes index
true = y_test.ravel() #this flattens the labels from (10000,1) -> (1,10000)

cm = confusion_matrix(true, pred)

plt.figure(figsize=(7,7))
plt.imshow(cm, cmap='Blues')
plt.xticks(range(10), labels, rotation= 45)
plt.yticks(range(10), labels)
plt.xlabel("Prediction label")
plt.ylabel("True label")
plt.title("Confusion Matrix")
plt.colorbar()
plt.tight_layout()
plt.show()

#classification_report will give us the following metrics: precision, recall, F1 score
print(classification_report(true, pred, target_names=labels))
```

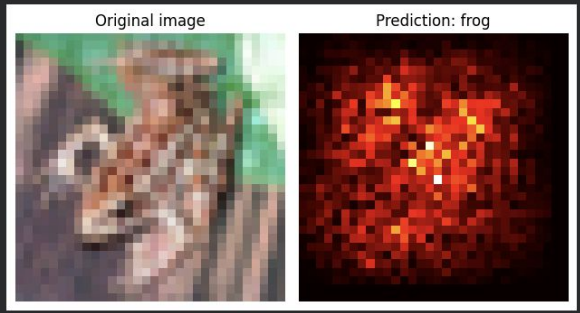



```
precision    recall  f1-score   support
```

- Shows how well the model predicts each class and where it gets confused.
- Highlights systematic errors between visually similar categories.

```python
#discover the important pixel in the image that helps the model make the prediction
import tensorflow as tf #saliency = going pixel level
def saliency(model,img):
  img_tensor = tf.convert_to_tensor(img[None], dtype=tf.float32)
  with tf.GradientTape() as tape: #track the pixel in the img_tensor
    tape.watch(img_tensor)
    preds = model(img_tensor,training = False)
    cls = tf.argmax(preds[0])
    class_pred = preds[:,cls]
  grads = tape.gradient(class_pred, img_tensor) [0]
  sal = tf.reduce_max(tf.abs(grads), axis=-1)
  sal = (sal - tf.reduce_min(sal)) / \
        (tf.reduce_max(sal) - tf.reduce_min(sal) + 1e-8)
  return sal.numpy(), int(cls.numpy())


#Choose a random number for an image from the dataset
i = 95
sal, cls_pred = saliency(model, x_test[i])

#ploting the image results
plt.subplot(1,2,1)
plt.imshow((x_test[
plt.title("Original image")
plt.axis('off')

plt.subplot(1,2,2)
plt.imshow(sal, cma
plt.title(f"Predict
plt.axis('off')
plt.tight_layout()
plt.show()
```

- Identifies which pixels most influenced the model's prediction.
- Explains why the model classified the image as a specific object, not just the result.



Thank You