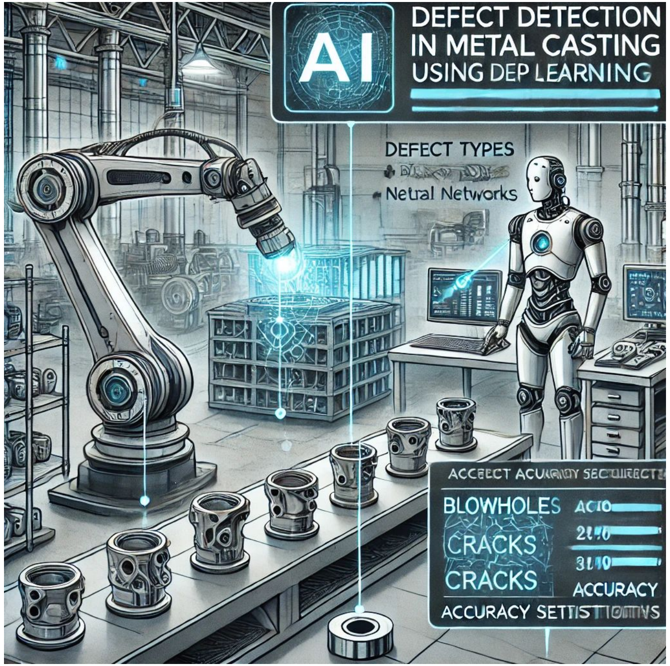
Springboard

AI Innovation in Quality Control:
Harnessing Deep Learning and Convolutional Neural Networks to detect Defective Metal Parts in Manufacturing environment

3rd Data Science Intensive Capstone Project
September 9th, 2024
By Javier Jorge Pérez Ontiveros

Executive Summary

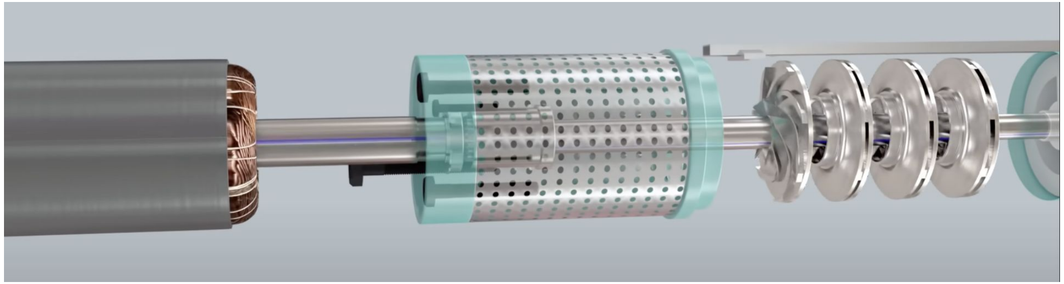
- This project used **AI and Deep Learning (CNNs)** to automate quality control in metal casting for submersible pump impellers, improving accuracy and efficiency over manual inspections



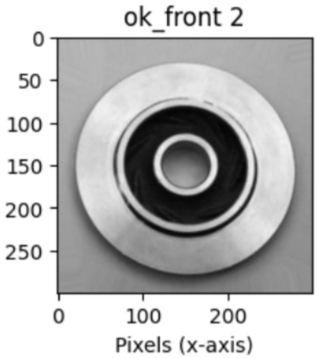
<u>Key Points:</u>

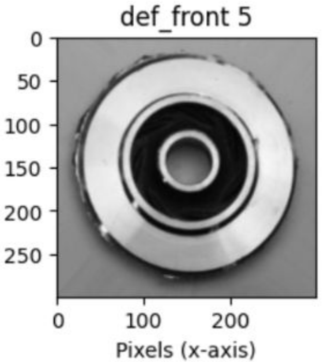
- **Objective:** Develop a CNN model to classify parts as “Defective” or “OK” with at least 95% accuracy and reduce inspection time by 50%
- **Results:**
  - A custom CNN model achieved **98% accuracy**
  - A transfer learning model (VGG16) achieved **99% accuracy** with faster implementation
  - The system can process up to **90,000 pieces/hour**, significantly improving efficiency
- **Conclusion:** Transfer learning proved more efficient for real-world deployment, offering high accuracy and saving time

Product Application



OK | Defective Part





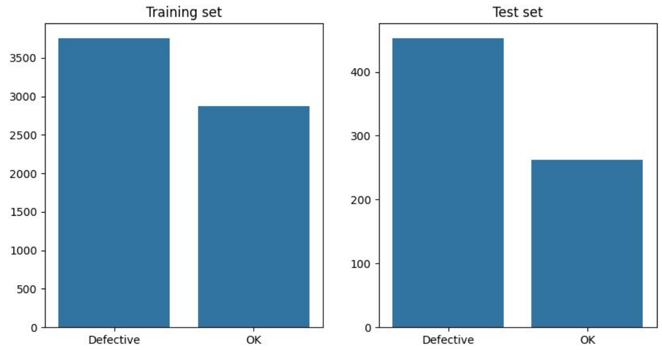
The main problem in manufacturing metal casting products for submersible pump impellers is **water leakage**

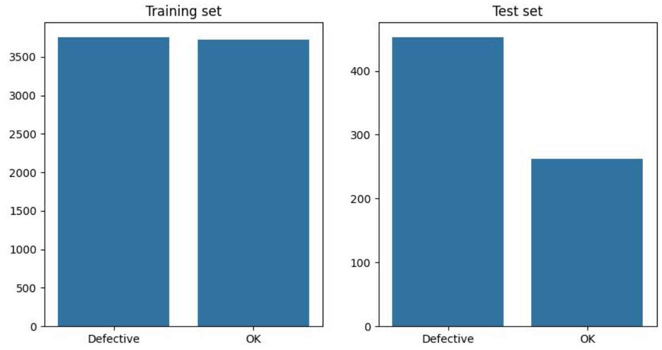
Detection takes a lot of resources: **time & labor**

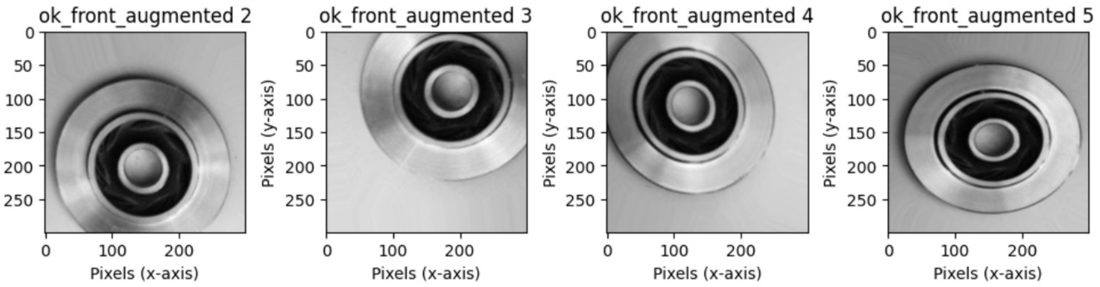
And can be inconsistent because of **human errors**

Defects like blow holes, pinholes, burrs, and shrinkage can cause significant **financial losses** if not detected correctly and on time

Data Wrangling

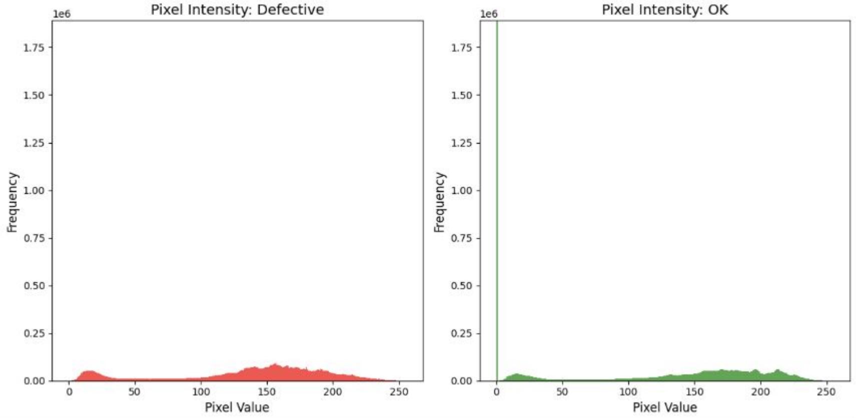


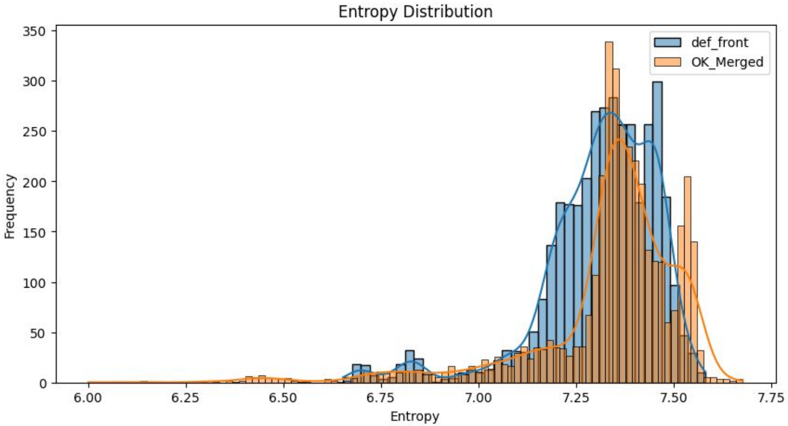




As the training set was imbalanced, Data Augmentation Techniques were utilized to generate new pictures scenarios: Rotation, Zoom, Flips, Brightness, etc

EDA





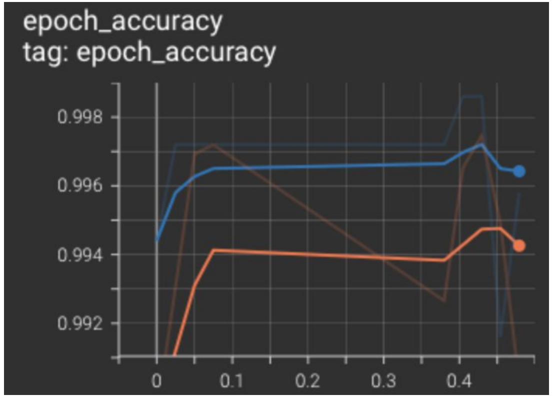
Pixel intensity was tested in both samples, showing a slight difference between the two groups, but the pattern is not easily identified by humans

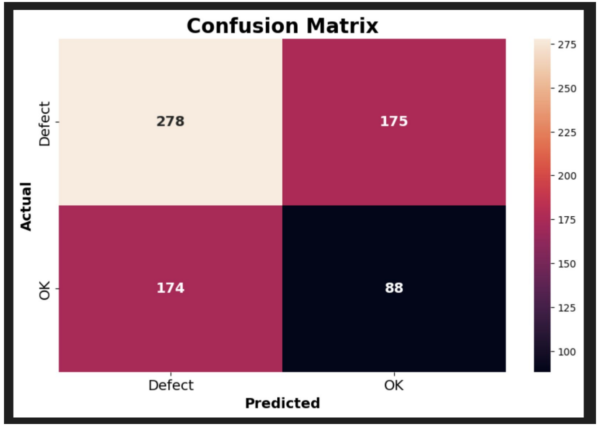
Entropy was also measured to identify differences between two categories of pictures by helping quantify the level of uncertainty or “disorder” in the image data

These 2 Analysis revealed differences in pixel intensity distribution, which could indicate unique characteristics in defective pumps

First CNN Model Creation

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 298, 298, 32)      896

 max_pooling2d (MaxPooling2  (None, 149, 149, 32)      0
 D)

 conv2d_1 (Conv2D)           (None, 147, 147, 64)      18496

 max_pooling2d_1 (MaxPoolin  (None, 73, 73, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 71, 71, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 35, 35, 128)       0
 g2D)

 flatten (Flatten)           (None, 156800)            0

 dense (Dense)               (None, 128)               20070528

 dropout (Dropout)           (None, 128)               0

...
Total params: 20163905 (76.92 MB)
Trainable params: 20163905 (76.92 MB)
Non-trainable params: 0 (0.00 Byte)
```



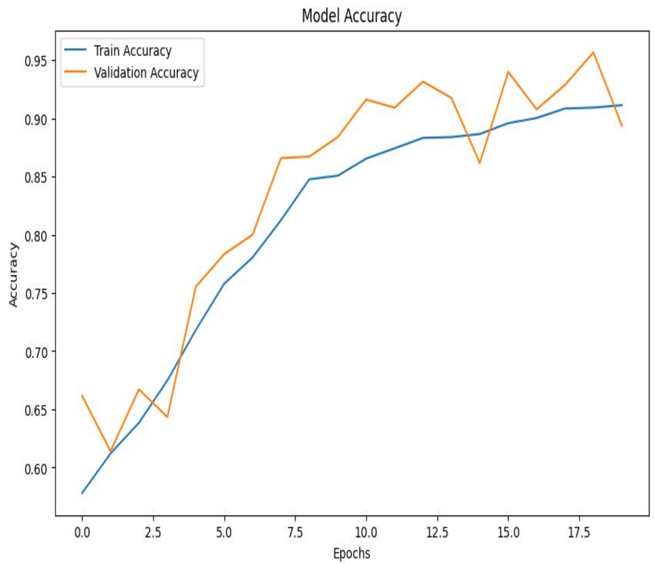


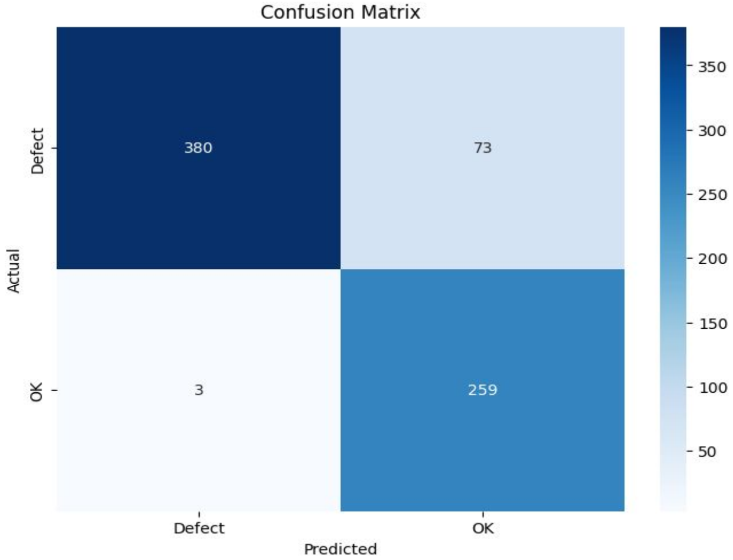
During Training the model achieved an 98% accuracy, however when it was tested on “unseen” data, this dropped quickly to **63% accuracy**

Second CNN Model

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 150, 150, 16)	2368
max_pooling2d (MaxPooling2D)	(None, 75, 75, 16)	0
conv2d_1 (Conv2D)	(None, 75, 75, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 37, 37, 32)	0
conv2d_2 (Conv2D)	(None, 37, 37, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 18, 18, 64)	0
flatten (Flatten)	(None, 20736)	0
dense (Dense)	(None, 224)	4645088
dropout (Dropout)	(None, 224)	0

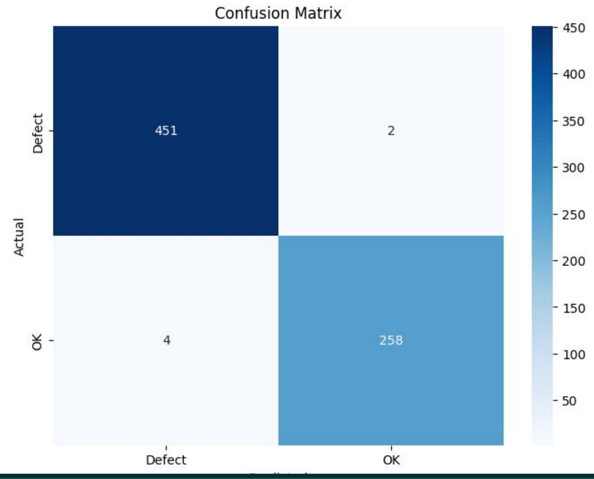


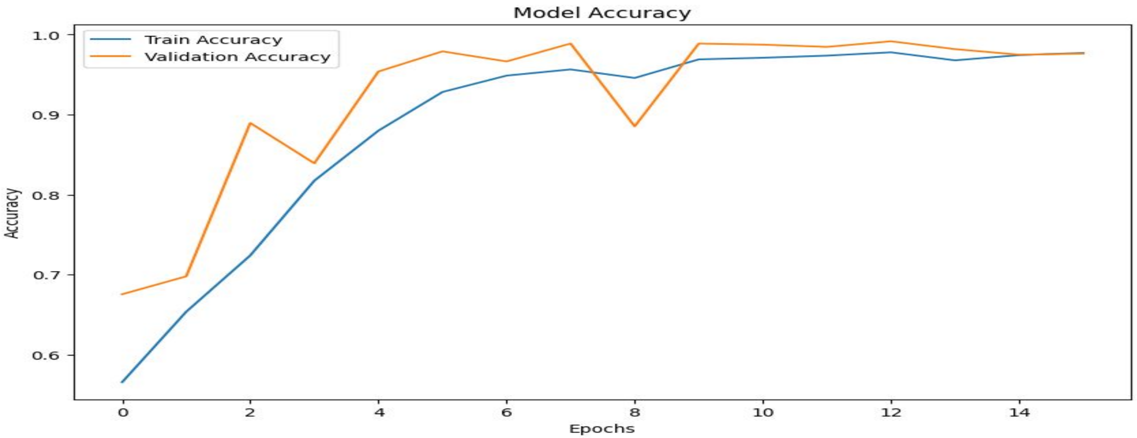


In this model, the Architecture was changed to a Simpler model design
Test Accuracy achieved was up to 89%

Hyperparameter Tunning

```
Best val_accuracy So Far: 0.988811194896698
Total elapsed time: 01h 33m 03s

The optimal number of filters for the first Conv2D layer is 16,
for the second Conv2D layer is 128,
for the third Conv2D layer is 256.
The optimal number of units in the Dense layer is 512.
The optimal dropout rate is 0.30000000000000004.
The optimal learning rate is 9.878314341240697e-05.
```



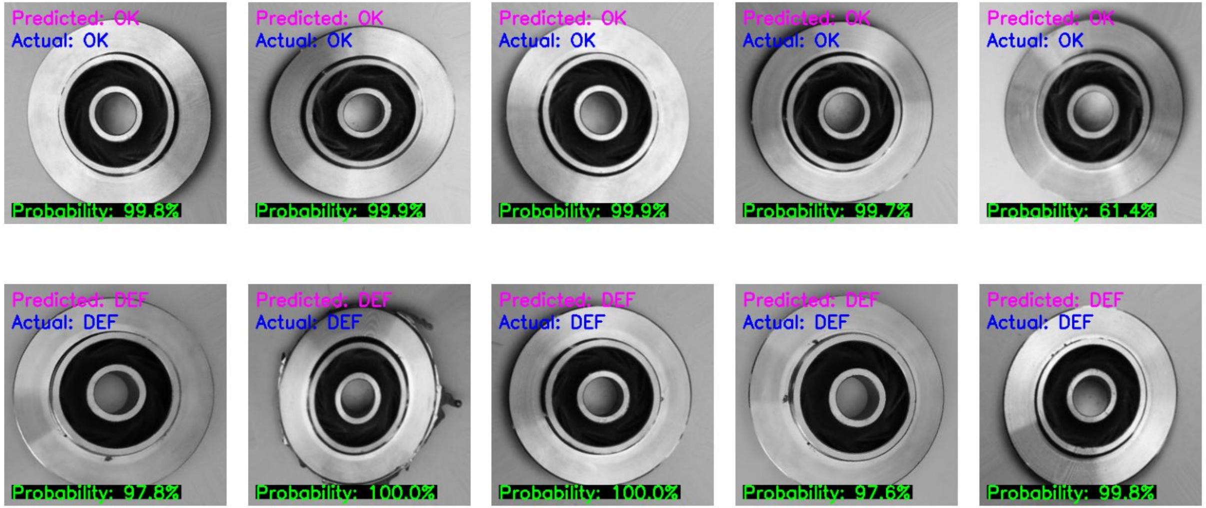


16 out of 20 epochs were compiled (Early stop was activated when the model stopped improving accuracy).

With this hyperparameter tunning the Test **accuracy improved > 98%**

Results and Evaluation of Second Model

This visual representation of 10 random pictures (5 OK & 5 Defective), the Actual Labels can be observed in **Blue** and Predicted Values in **Pink**.



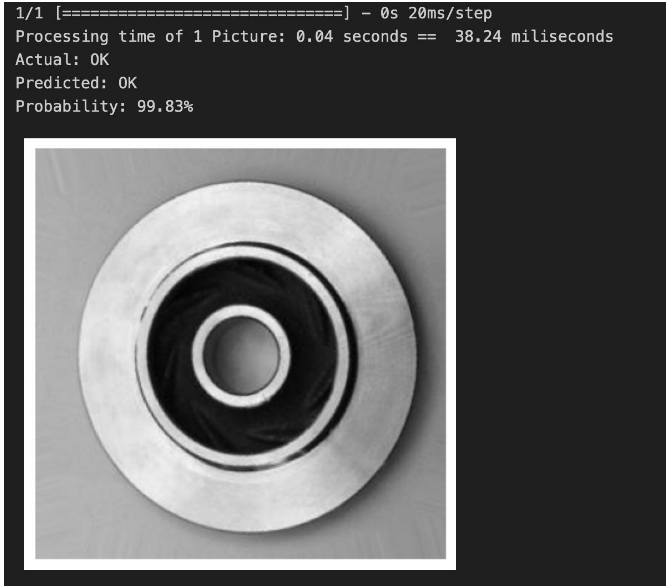
The probability (Green), is obtained from the Sigmoid Activation Function in the last layer of the model.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

All pictures of the random sample were correctly classified

Probability % represents how certain is the model of the decision taken

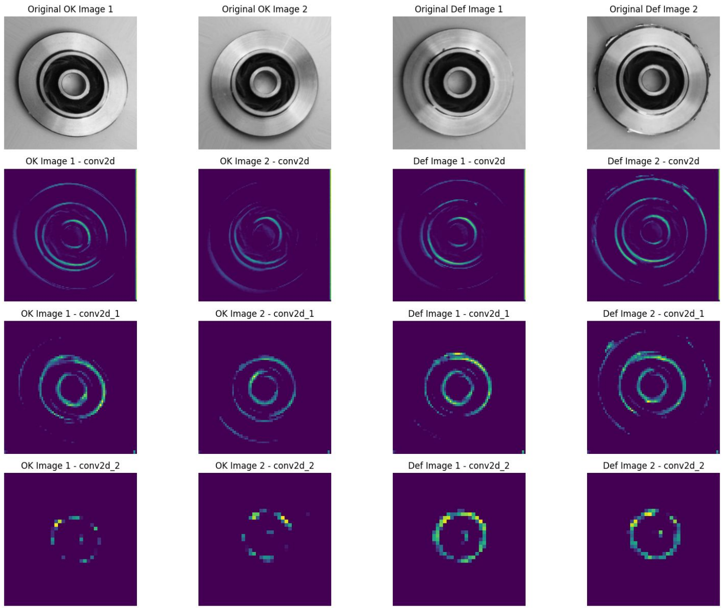
Processing Time of 1 Part

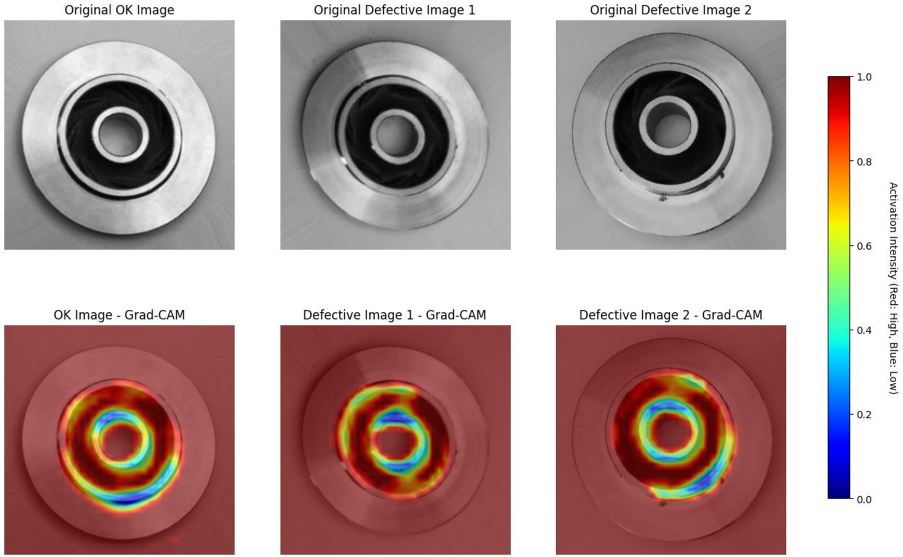


A computer clock was stablished before and after running trials, and the best processing time was **0.04 seconds per piece.**

This means that the Neural Network could analize **up to 90,000 pcs/ hr!**

How the model 'see' the defects

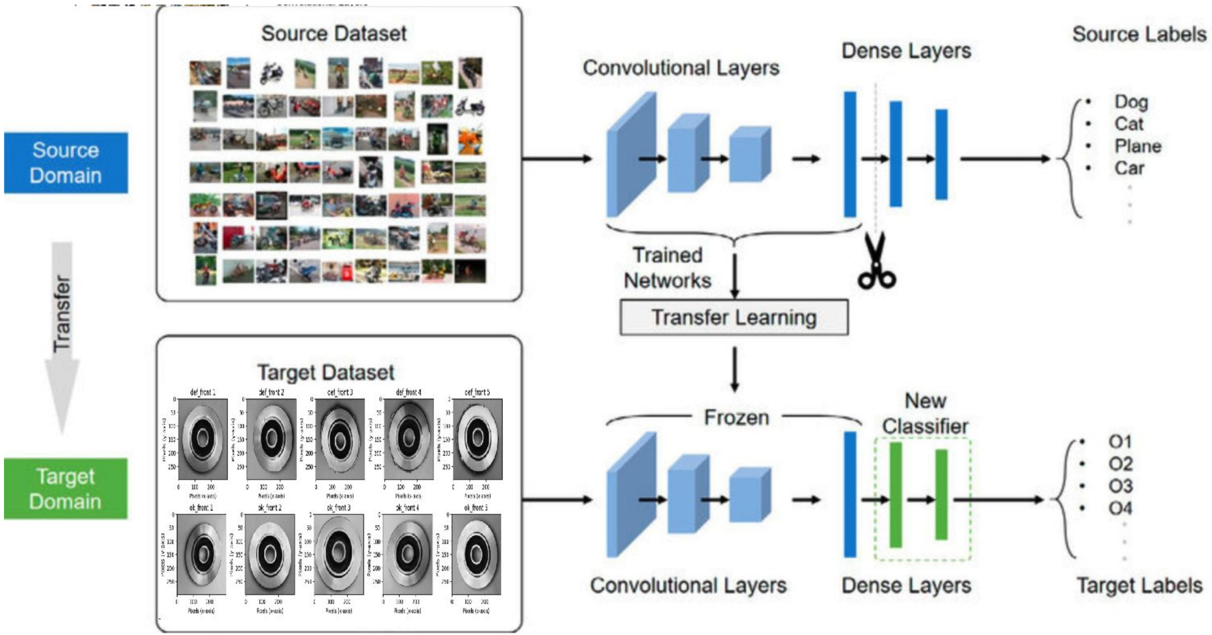




Using Keras Model and Image modules, the outcome of the 3 layers of Convolutional Neural Networks was visualized to see how the model "captures" the defects.

Using Grad-Cam combined with Tensor flow, the pixel Activation Intensity was visualized. Red Areas are zones that are highly considered by the model, while blue areas are areas that the computer somehow "ignores".

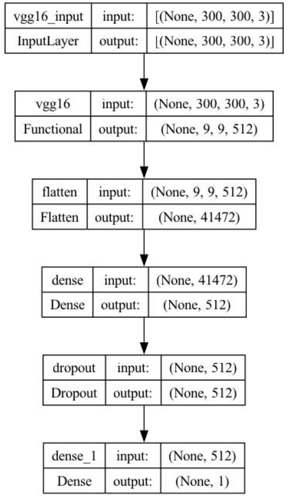
Transfer Learning



- **Transfer Learning** reuses pre-trained models **(e.g., ResNet, VGG)** that have learned to detect general visual features like edges and textures.
- **VGG16**, pre-trained on ImageNet (14 Million images, 1,000 categories), **was fine-tuned for** defect detection in **metal casting.**
- In this project, the first 13 convolutional layers were used as feature extractors, and custom dense/dropout layers were added for binary classification

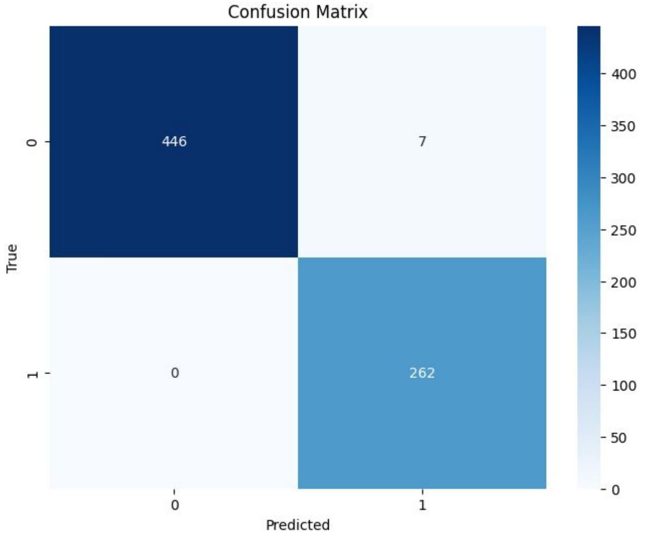
Transfer Learning reduces resources needed and improves model focus on relevant features while preventing overfitting

Transfer Learning Performance




```
Found 7487 images belonging to 2 classes.
Found 715 images belonging to 2 classes.
Epoch 1/10
234/234 [==============================] - 477s 2s/step - loss: 0.5017 - accuracy: 0.7711 - val_loss: 0.2010 - val_accuracy: 0.9455
Epoch 2/10
234/234 [==============================] - 510s 2s/step - loss: 0.2284 - accuracy: 0.9193 - val_loss: 0.0969 - val_accuracy: 0.9888
Epoch 3/10
234/234 [==============================] - 496s 2s/step - loss: 0.1477 - accuracy: 0.9545 - val_loss: 0.0776 - val_accuracy: 0.9888
Epoch 4/10
234/234 [==============================] - 494s 2s/step - loss: 0.1141 - accuracy: 0.9671 - val_loss: 0.0512 - val_accuracy: 0.9916
Epoch 5/10
234/234 [==============================] - 472s 2s/step - loss: 0.1078 - accuracy: 0.9662 - val_loss: 0.0524 - val_accuracy: 0.9902
Epoch 6/10
234/234 [==============================] - 470s 2s/step - loss: 0.0900 - accuracy: 0.9729 - val_loss: 0.0368 - val_accuracy: 0.9930
Epoch 7/10
234/234 [==============================] - 467s 2s/step - loss: 0.0845 - accuracy: 0.9730 - val_loss: 0.0360 - val_accuracy: 0.9930
Epoch 8/10
234/234 [==============================] - 466s 2s/step - loss: 0.0759 - accuracy: 0.9773 - val_loss: 0.0367 - val_accuracy: 0.9930
Epoch 9/10
234/234 [==============================] - 468s 2s/step - loss: 0.0741 - accuracy: 0.9774 - val_loss: 0.0541 - val_accuracy: 0.9902
Epoch 10/10
234/234 [==============================] - 468s 2s/step - loss: 0.0609 - accuracy: 0.9829 - val_loss: 0.0889 - val_accuracy: 0.9804
23/23 [==============================] - 41s 2s/step - loss: 0.0360 - accuracy: 0.9930
Test Loss: 0.03601987287402153
Test Accuracy: 0.9930070042610168
```

Overall: 99.3% Accuracy!



The Transfer Learning model achieved a high accuracy (99%) since the 4th Epoch

Further modification of its layers wasn't necessary

Model Selection

While building a **model from scratch was a good exercise** to practice the concepts learned during the Data Science Certification.

If I had to start this project again**, I would select Transfer Learning** method as a starting point due to its simplicity for implementation.

Conclusion

The **model from scratch reached 98% accuracy,** showing that it could learn and generalize the key features needed to classify metal parts. Initially, the model was too complex and overfitted, but after simplifying it, it worked well.
But **Transfer Learning model** extremely well with **99% accuracy, with the benefit of saving a lot of time and resources,** since the pre-trained model already had a good understanding of general visual features.

Thank you!

https://www.linkedin.com/in/jjpo/

javierjorge77@gmail.com

https://github.com/javierjorge77/Springboard/tree/main/Capstones/Capstone3_backup

(+521) 8711777903