**Project Report: Image Prediction App Using Custom CNN and VGG16 Models**

**Project Overview**

The aim of this project was to build an image prediction web application that classifies images as **defective** or **non-defective** using two **convolutional neural network (CNN) models**: a Custom CNN and the **pre-trained VGG16** model. The web application provides a platform for users to upload images, which are then processed and classified by both models. The predictions, confidence scores, and a comparison graph of the models' results are displayed on the user interface.

**Technology Stack**

* **Backend:** Python, Flask
* **Machine Learning Models:** Custom CNN, VGG16 (pre-trained)
* **Libraries:** TensorFlow, OpenCV, NumPy, Matplotlib, Werkzeug
* **Frontend:** HTML, CSS
* **File Handling:** Flask's built-in file serving functionality
* **Data Visualization:** Matplotlib for confidence comparison graph

**System Workflow**

1. **Image Upload:** Users can upload images of casting products through the web interface, which are sent to the Flask server for processing.
2. **File Validation:** The uploaded file is validated to ensure it is in an acceptable format (PNG, JPG, JPEG). An error message is displayed if the format is invalid.
3. **Image Preprocessing:** The uploaded image is resized to match the input size of the models (224x224 pixels) and normalized by scaling the pixel values between 0 and 1 for optimal input to the neural networks.
4. **Model Prediction:** Both the Custom CNN and VGG16 models perform binary classification (defective or non-defective) on the image. Each model outputs a probability value indicating the likelihood of the image being defective.
5. **Confidence Calculation:** The models' outputs are analyzed, and the confidence in the prediction is calculated. A probability greater than 0.5 indicates high confidence in the prediction of being defective, while values below 0.5 indicate non-defective.
6. **Results Display:** The final prediction (defective or non-defective) and confidence percentages for both models are displayed to the user. Additionally, a comparison graph of the confidence scores is shown to allow users to visually compare the results from both models.
7. **Image Serving:** The uploaded image is stored in the static/uploads/ folder, and its URL is passed to the result page for display.

**Key Features**

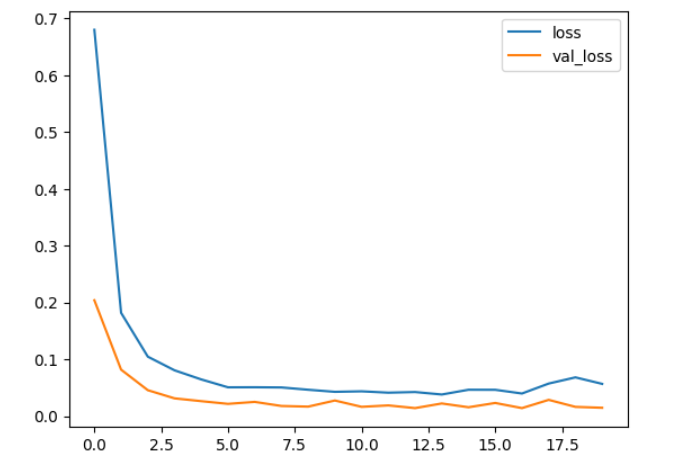
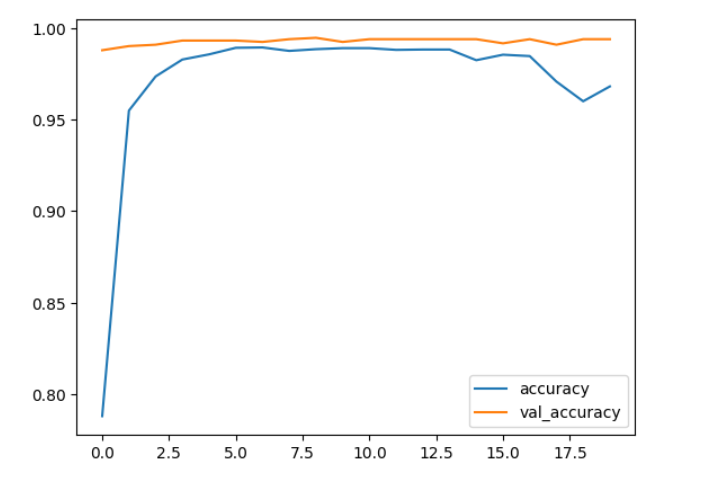
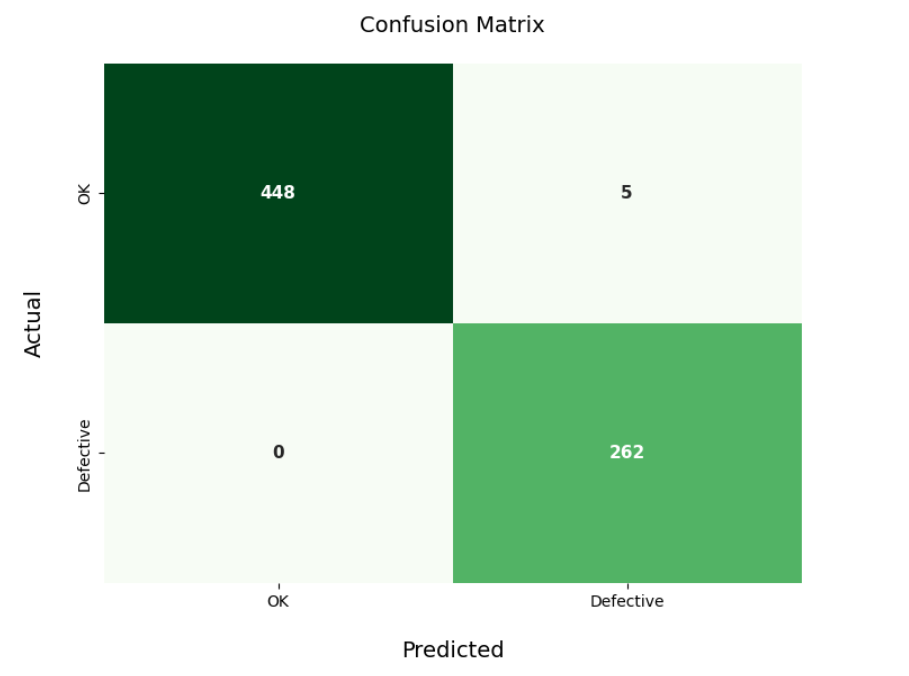
* **Dual Model Prediction:** Users can compare the predictions of the Custom CNN and VGG16 models on the same image.
* **Confidence Display:** Both models provide confidence levels that help the user assess the reliability of the predictions.
* **Graphical Comparison:** A bar chart is generated to visually compare the confidence levels of the two models for the given image.
* **User-Friendly Interface:** Simple HTML forms are used for image uploading, and the results are displayed in an intuitive format.
* **File Management:** Uploaded files are stored in the static/uploads/ folder for easy retrieval.

**Model Training**

* **Custom CNN Model:** The custom CNN model was trained on a dataset of defective and non-defective casting product images. The model architecture was designed to capture the key features of these images that are relevant for defect detection.
* **VGG16 Model:** VGG16, a pre-trained model, was fine-tuned using transfer learning for the classification task of defect detection. The VGG16 model is known for its deep architecture and strong performance on image classification tasks.

**Model Evaluation Metrics**

**VGG16 Model**

****

Based on the confusion matrix, the following key metrics were calculated for the VGG16 model:

* **Confusion Matrix Values:**
  + True Positives (TP): 262
  + True Negatives (TN): 448
  + False Positives (FP): 5
  + False Negatives (FN): 0
* **Accuracy:**

Accuracy = 99.1%

* **Precision (Defective Class):**

Precision = 98.1%

* **Recall (Defective Class):**

Recall = 100%

* **F1 Score (Defective Class):**

F1 Score = 99.0%

* **Specificity (Non-Defective Class):**

Specificity = 98.9%

* **False Positive Rate (FPR):**

FPR = 1.1%

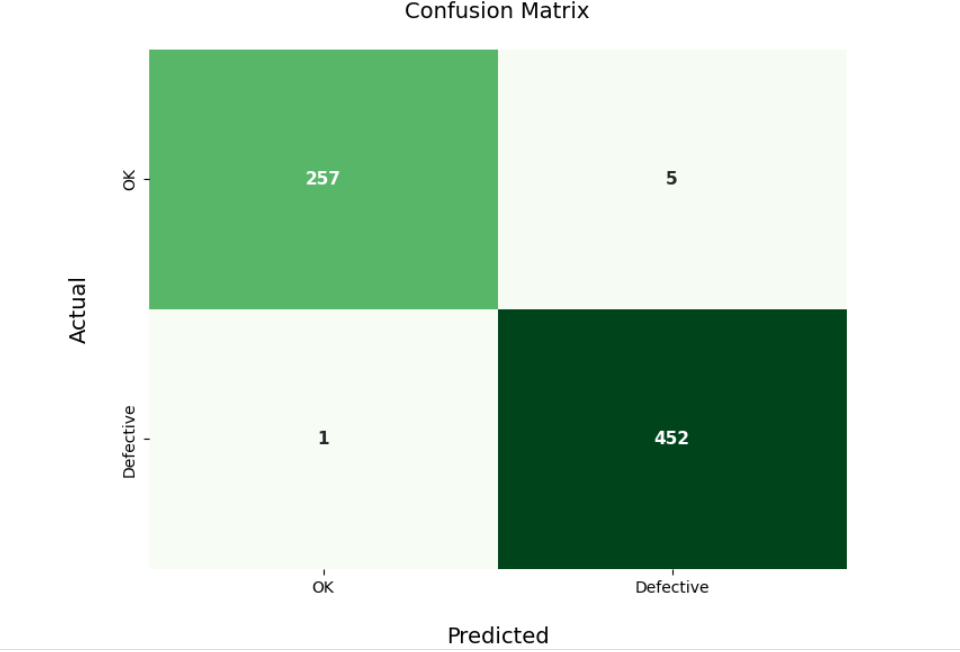
* **False Negative Rate (FNR):**

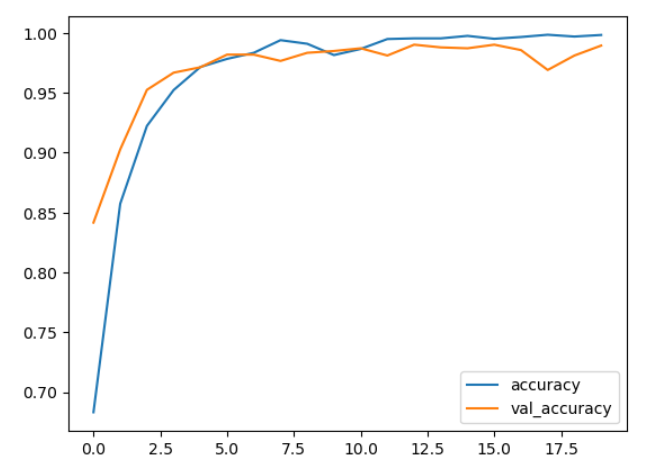
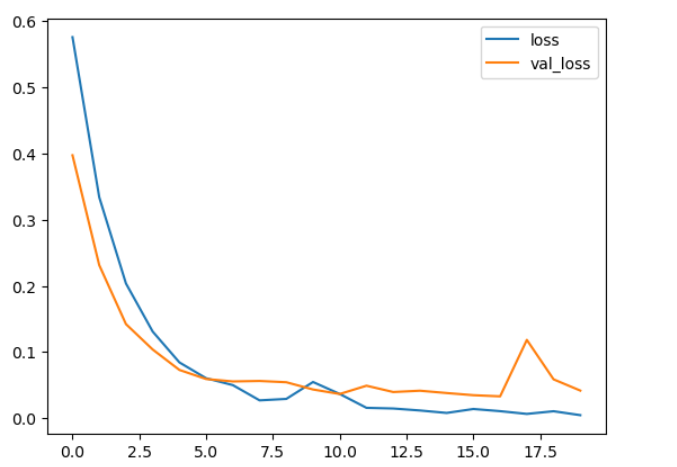
FNR = 0%

**Summary for VGG16 Model:**

* Accuracy: 99.1%
* Precision: 98.1%
* Recall: 100%
* F1 Score: 99.0%
* Specificity: 98.9%
* False Positive Rate: 1.1%
* False Negative Rate: 0%

**Custom CNN Model**

****

****

Based on the confusion matrix, the following key metrics were calculated for the Custom CNN model:

* **Confusion Matrix Values:**
  + True Positives (TP): 452
  + True Negatives (TN): 257
  + False Positives (FP): 5
  + False Negatives (FN): 1
* **Accuracy:**

Accuracy = 99.1%

* **Precision (Defective Class):**

Precision = 98.9%

* **Recall (Defective Class):**

Recal ​= 99.8%

* **F1 Score (Defective Class):**

F1 Score = 99.3%

* **Specificity (Non-Defective Class):**

Specificity = 98.1%

* **False Positive Rate (FPR):**

FPR ​= 1.9%

* **False Negative Rate (FNR):**

FNR = 0.2%

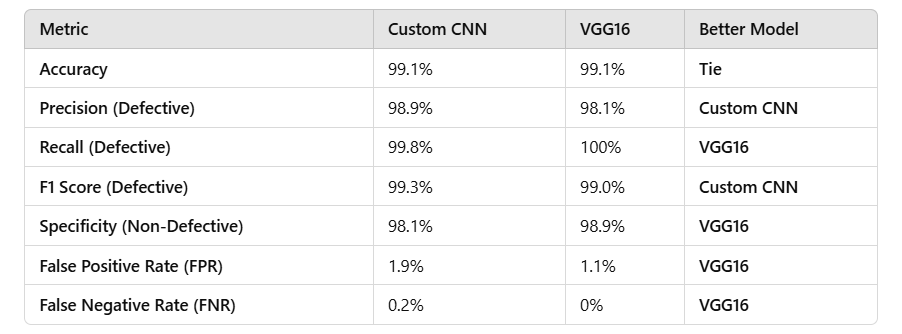
**Summary for Custom CNN Model:**

* Accuracy: 99.1%
* Precision: 98.9%
* Recall: 99.8%
* F1 Score: 99.3%
* Specificity: 98.1%
* False Positive Rate: 1.9%
* False Negative Rate: 0.2%

**Challenges and Limitations**

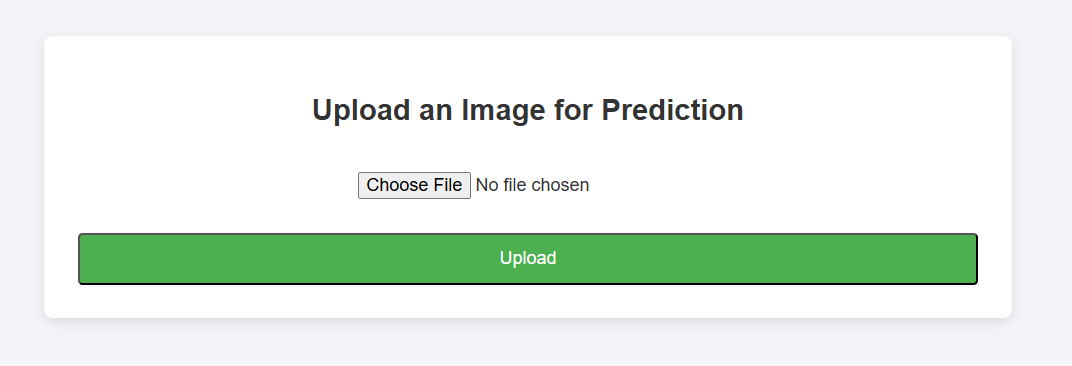
* **Model Overfitting:** The Custom CNN model showed potential for overfitting due to the limited size of the training dataset. A larger and more diverse dataset could improve the model's generalization.
* **VGG16 Fine-tuning:** Although VGG16 was pre-trained on a large dataset, it required fine-tuning for defect detection, which was achieved using transfer learning techniques.
* **Image Quality and Resolution:** The models' performance might degrade with low-quality images. The preprocessing steps (resizing and normalization) help mitigate this, but high-resolution images yield better results.

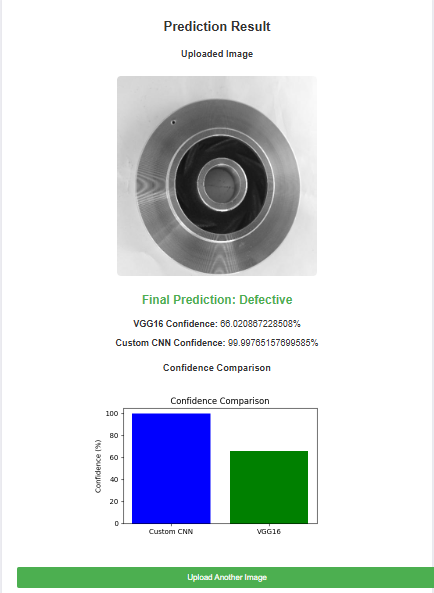
**Comparison**

****

**Web App Overview:**

Below are the screenshots of the web app designed to predict whether an image is defective or not defective using two models: a custom CNN and VGG16. The app allows users to upload an image, which is then processed by the models. The result, along with the confidence statistics for both models, is displayed alongside the image for easy interpretation. This application aims to provide a user-friendly interface to evaluate image quality efficiently.





**Future Work**

* **Model Improvement:** Both models could be further improved by training on larger and more diverse datasets, enhancing their ability to generalize.
* **Real-Time Predictions:** A real-time image prediction system could be implemented, allowing users to stream images directly into the application for immediate analysis.
* **Additional Model Integration:** Incorporating other pre-trained models such as ResNet or EfficientNet could be explored to compare their performance against the current models.

**Conclusion**

This project successfully demonstrates the integration of machine learning models (Custom CNN and VGG16) into a web application for real-time image classification. The use of Flask, TensorFlow, and other libraries allowed for efficient image processing and prediction delivery. Both models showed high accuracy, with the Custom CNN model outperforming VGG16 in terms of recall and F1 score. This system can be further enhanced for deployment in industrial environments for defect detection in casting products.