MANE 4962 Final Report

Meat Labeler

Stephen Feldman

4/16/2025

Contents

[Executive Summary 4](#_Toc195361811)

[Introduction 6](#_Toc195361812)

[Project Overview 6](#_Toc195361813)

[Literature Review 6](#_Toc195361814)

[Problem Definition 8](#_Toc195361815)

[Methods and Procedure 9](#_Toc195361816)

[Overview 9](#_Toc195361817)

[Data Processing 9](#_Toc195361818)

[Simple CNN 9](#_Toc195361819)

[XGBoost 10](#_Toc195361820)

[ViT Classifier 11](#_Toc195361821)

[Masked CNN 12](#_Toc195361822)

[Dataset and Visualization 13](#_Toc195361823)

[Dataset Summary and Statistics 13](#_Toc195361824)

[Dimensionality Reduction with PCA and Visual Trends 13](#_Toc195361825)

[Pixel Intensity Histograms and Per-Class Visualization 14](#_Toc195361826)

[Results and Discussion 17](#_Toc195361827)

[Individual Model Breakdown 17](#_Toc195361828)

[Simple CNN 17](#_Toc195361829)

[XGBoost Classifier 21](#_Toc195361830)

[ViT Model 25](#_Toc195361831)

[Masked CNN 28](#_Toc195361832)

[Comparative Results 33](#_Toc195361833)

[Conclusion 34](#_Toc195361834)

[Data and Code 35](#_Toc195361835)

[References 36](#_Toc195361836)

Table 1: Count by class of images of meat 13

Table 2: Simple CNN Model Information 17

Table 3: Simple CNN Layers with Tuned Best Hyperparameters 18

Table 4: Simple CNN Performance Metrics 18

Table 5: Simple CNN Classification Report 18

Table 6: Optuna Results for XGBoost Tuning 21

Table 7: XGBoost Classification Report 22

Table 8: XGBoost Model Metrics 23

Table 9: ViT Classification Report 25

Table 10: ViT Model Performance Metrics 26

Table 11: Masked CNN Classification Report 29

Table 12: Masked CNN Performance Summary 29

Table 13: Comparison of Model Performance Across All Classifiers 33

Figure 1: PCA Plot of Principal Components 14

Figure 2: Overlayed pixel value histogram for fresh images 15

Figure 3: Overlayed pixel value histogram for half-fresh images 15

Figure 4: Overlayed pixel value histogram for spoiled images 15

Figure 5: Simple CNN Confusion Matrix 19

Figure 6: Simple CNN Precision Recall Curve by Class 20

Figure 7: Simple CNN ROC Curves by Class 20

Figure 8: XGBoost Confusion Matrix 23

Figure 9: XGBoost Precision Recall Curve by Class 24

Figure 10: XBGoost ROC Curve by Class 24

Figure 11: ViT Confusion Matrix 27

Figure 12: ViT Precision-Recall Curves per Class 27

Figure 13: ViT ROC Curves Per Class 28

Figure 14: Feature Screening Masks For Masked CNN 30

Figure 15: Masked CNN Confusion Matrix 31

Figure 16: Masked CNN Precision-recall Curves per Class 31

Figure 17: ROC Curves Per Class 32

# Executive Summary

Continuous monitoring enables a transformation of quality control methodologies and corresponding process improvements. Manufacturers that work with large volumes currently employee statistical process control to protect themselves and consumers. These methods, while effective, are costly due to their reliance on human data collection. Reliability comes at the price of increased sampling, which requires slowing production are hiring more staff. As a result, employees spend valuable time doing routine quality checks in manufacturing and commercial settings. The recent trend in manufacturing has been the integration of bigdata and IoT strategies into their quality control framework. This offers quality built directly into the process where the quality of the equipment is monitored, reducing costs. However, certain industries struggle with implementing automated approaches because the assessment of quality is more complicated than a simple measurement which can be taken by a machine. Whereas a bolt has measurements and qualities that are explicitly measurable, food, in particular meat, can spoil. Detecting this spoiled meat is critical to brand perception and maximizing valuable product.

This project aims to begin developing a strategy to scan all meat entering the production process to mitigate costs associated with allowing spoiled meat to continue through the packaging and subsequent shipping or storage costs. Successful detection of spoiled meat visually will prove the use of visual quality control at scale can be used to monitor entire production lines. The project utilizes a publicly available dataset of meat which has been labeled according to its type (fresh, half-fresh, spoiled) for each picture. Mimicking a production setting the pictures are taken from approximately the same angle with slight variation. This parallels a production setting where a camera would be fixed above a production line with some variability in the placement of meat in the scope. This public dataset is used to train 4-different machine learning models. There is a clear difference in coloration between meats of different types which supported the 3-main model types to analyze the images: CNN, ViT, XGBoost. The final model is generated by applying a interpretability screening to the incoming data and training a new CNN on the screened data.

To make a recommendation with regards to what model to use, a comparison of model accuracy, size, and the probability of misidentified spoiled meat (risk to consumer), were used. The XGBoost model benefited from having the smallest storage size, moderate accuracy, and the lowest processing speed as it was working with images flattened to 32 by 32, but it is let down by its high probability of identifying spoiled meat as one of the types that is not spoiled. The ViT, performs decidedly average, with a size comparable to the CNN, a similar processing speed. Where the transformer shines is with the accuracy which just above 80% and the phenomenally low rate of misclassification of spoiled meats just under 7% which half the next best result obtained by the base CNN. The base CNN scored the highest accuracy, and second lowest probability of misidentification of spoiled meat, but had the highest model size and comparable processing speed to the ViT but by using 224 by 224 images it is worse compared to the XGBoost. The CNN incorporating a interpretability screening performed worse than the any other model across all categories, likely cause by additional tuning being needed in the interpretability algorithm. The created mask proved too strong, however, the insights generated from the current masking algorithm can generate insights with regards to which areas of the meat cut are most likely to spoil first. Analysis of the results shows that most of the models struggle with distinguishing the category of half-fresh meat. This makes sense at this meat is typically left out for a moderate period of time and can have features of spoiled and non-spoiled meat. This challenge was identified in the preliminary data analysis and warrants an additional feature being used such as the age of the meat to help guide the definition of the category in future exploration. The ViT proved the most capable of defining the categories of spoiled and fresh which in terms of production monitoring makes it the most valuable model.

Based on the results of model testing there is evidence to conclude that machine learning models are ready to be used to replace human visual screenings for the purposes of quality control, or to significantly reduce the need for screenings. However, the choice of the classifier type employed in industry should vary by local context. Through tuning and skillful formulation, the models offer similar overall performance, and selection should be driven by specific requirements. Given a purely visual screening, the transformer combined with some light masking to hide the background of the meat may prove to be the most effective strategy for delivering a solution. Installing a fixed camera and allocating an employee for data labeling would start the implementation process culminating with a trained model able to operate autonomously, communicating directly with the process logic controller.

# Introduction

## Project Overview

Applying automated quality control is critical to improving the efficiency of manufacturing processes. Current strategies require significant sampling rates combined with statistical methods to manage process quality, these strategies are costly and often are implemented outside of process improvement frameworks. Automating the process enables the scanning of all products in the process, making decisions in real time based on product quality, and data for future process improvement projects. Processed food in particular is impacted by the need to perform many quality checks (most of which are handled visually) while also producing at very large scales. Delays in catching quality issues that are measured in minutes can still cost hundreds of pounds of lost product. Personal experience working on Goldfish crackers further supports the need for continuous data collection and automated analysis.

This project serves as a proof of concept to demonstrate that classification of food products at scale can be accomplished with a high degree of accuracy and reasonable resources to be implemented at scale. By comparing and analyzing different classification methods it will be possible to determine what framework to use when implementing the image classification for a specific local process. Success is determined by driving the models to an accuracy level that is in excess of 75% on the validation data while trying to optimize for the lowest probability of misclassifying spoiled meat in either of the safe to eat categories. The modest accuracy target is set due to challenges with the half-fresh category which will be discussed further in the data section. In turn, this proves that machine learning models can be used to measure meat quality and ensure consumer safety.

To accomplish this, a full end-to-end image classification pipeline was developed, incorporating data preprocessing, model design, training, evaluation, and comparative analysis. Four distinct classification frameworks were implemented: a standard Convolutional Neural Network (CNN), a modified CNN using interpretable binary masking, a tabular-based XGBoost classifier, and a Vision Transformer (ViT). Each model was optimized using Optuna for hyperparameter tuning and evaluated across a common dataset of meat images labeled by freshness class. These experiments were designed to simulate realistic manufacturing constraints, including the need for real-time predictions, minimal misclassification of unsafe product, and deployment feasibility based on model size and complexity.

Throughout the project, key performance indicators such as classification accuracy, precision-recall balance, confusion matrix behavior, and ROC AUC were used to assess each model. Emphasis was placed on correctly identifying spoiled products and minimizing false negatives in that category, as these pose the greatest food safety risk. In addition to evaluating performance, the project explored interpretability through saliency-based masking and feasibility through model size comparisons. The result is a set of recommendations for manufacturers seeking to integrate AI-driven visual inspection systems into their production lines. Further targeted experimentation is slated to produce a very robust model for quality visual inspection of meat, and most other similar food processing settings.

## Literature Review

Recent advances in computer vision have enabled non-destructive, real-time assessment of meat freshness using deep learning. Song et al. (2024) demonstrated that a lightweight MobileNetV3\_Small model, when combined with attention mechanisms, can achieve over 98% accuracy in classifying pork cuts and freshness, highlighting the feasibility of deploying CNNs on production lines with limited computational resources (Song et al., 2024). Elangovan et al. (2024) further showed that an ensemble of compact CNNs (ConvNet‑18 and ConvNet‑24) yields comparable performance (up to 99.4% accuracy) while reducing model complexity, motivating our exploration of both single‑model and ensemble frameworks (Elangovan et al., 2024).

Beyond CNNs, transformer‑based architectures have emerged as powerful alternatives for image classification. Dosovitskiy’s work introduced the Vision Transformer (ViT), which leverages patch embeddings and self‑attention to capture global context, outperforming CNNs on several benchmarks and inspiring our adaptation of ViT for meat image classification (Dosovitskiy et al., 2021). Conversely, tree‑based methods such as XGBoost can offer competitive accuracy with minimal model size. Chen and Guestrin showed that gradient boosting on flattened pixel features can achieve strong results on image tasks, an approach that may be warranted by the resource constrained setting of consumer goods manufacturing (Chen & Guestrin, 2016).

Interpretability methods are critical for understanding model decisions and filtering input data. Ribeiro et al. (2016) proposed LIME to generate local saliency maps, which was combined with Random Forest importance scores to create binary masks that focus CNN training on relevant regions (Ribeiro et al., 2016). Although the CNN underperformed due to overly aggressive masking this approach underscores the potential of interpretability‑driven preprocessing. Finally, robust model training relies on effective hyperparameter tuning. While not detailed in prior meat‑specific studies, general frameworks like Optuna (Akiba et al., 2019) have proven essential for optimizing deep learning models. The Optuna package would be used to tune all 4 of the models that were created for this project.

These resources highlight potential avenues to pursue for tackling the challenges of meat classification and automated quality control in manufacturing settings. However, choosing the best framework for the task and generating recommendations for when to use each, especially in the processed sector, will assist industrial engineers in training local models. These models can also serve the auxiliary purpose of driving investment into data management tools and collection in turn paving the road for increased automation.

# Problem Definition

Machine learning is required to solve the problem of controlling quality of processed food because of the large volume of product flowing through the system, and due to the complexity of food quality. There is no set indicator for meat being spoiled, and thus far there is not a definitive sensor that can be used to prove that meat is spoiled that works within a second. By analyzing the visual frame, egregious cases of spoiled meat can be quickly eliminated before they get far into the production process without the need to have a human observer watching the line. An algorithmic approach is not sufficient to analyze the images due to the lack of one specific factor being the determinant between spoiled and unspoiled meat. Other factors such as smell combined with sight could be more effective but measuring smell in industrial conditions is technically challenging. The manufacturing environment also imposes the limitations of model size and image processing speed. A human can scan meat a limited rate, increasing the production rate without hiring more personnel requires a digitized/automated approach.

A typical project scope would imply the creation and implementation of the hardware on a production line, however, due to the costs and additional logistics associated with that this project is aiming to make a recommendation with regards to machine learning classifiers that could be used in such a product rather than create the product. The same applies to the data collection. As resources are limited the dataset will not be coming from a manufacturing environment and as a result may not result in a model that will be applicable out of the gate. Finally, limited computational resources led to a limited model complexity. This is reflective of the state of processed food manufacturing where resources are more limited than in precision manufacturing.

# Methods and Procedure

## Overview

The code for the project would first vectorize the image data and merge it with the corresponding label to create a dataset suitable for model creation. 4 separate python modules would train 4 different classifiers: CNN, XGBoost, ViT, pre-screened CNN. Each of these tuned models would be saved in the results folder where they would be analyzed in the corresponding analysis module. The results of each evaluator module were also saved in the results folder to be later accessed by results notebook. The final results notebook summarizes the results and compares all 4 models to one another enabling observers to make a determination with regards to the best one.

## Data Processing

The dataset from Kaggle contains 2 folders: a training folder and a validation folder. Within each folder this is an excel file containing the image name and the corresponding label (fresh, half-fresh, spoiled), and a set of images. To create a machine learning model the data must be converted to vector form matched to the corresponding labels. The very first script finds the downloaded Kaggle dataset then flatten and merge the images with their corresponding labels. The images were resized in 224 by 224 pixels with each pixel stored a numerical value representing the intensity of light in each pixel of red, blue, and green. Using the image file name as the key these new matrices are merged with the label from the spreadsheet. Following that process a number of data checks are built into the script to verify that the new dataset is complete. Such as checking for missing data and others. Finally, the script saves the data to the data folder with two .npy files containing the training and validation data, respectively. By compartmentalizing a group of the data for validation purposes it would ensure the accuracy of the model would be assessed fairly.

## Simple CNN

In the training module, the process sets random seeds to ensure reproducibility and checks if a GPU is available for training to verify the CUDA installations are working as expected. The code loads the preprocessed image data from the data folder where it was stored as .npy files after the data processing script was executed earlier. The script first takes the training data and does a randomized split to create validation data which would guide model optimization. The training module uses PyTorch to define a simple CNN with sequential convolutional layers interleaved with ReLU activations and max-pooling operations, followed by a flattening step, fully connected layers, and dropout for regularization. The dataset is first rearranged from the standard HxWxC format to the CxHxW format. The model then applies a series of three convolutional blocks. In the first block, a convolution layer with a 3x3 kernel processes the input using a tunable number of filters, followed by a ReLU activation to introduce non-linearity and max pooling layer with a 2x2 kernel to reduce spatial dimensions. The 2nd block is similar but uses a tunable number of filters for a 3x3 convolution again followed by ReLU activation and max pooling. The third block upgrades the feature depth by using 128 filters in its convolution layer with a 3x3 kernal followed by ReLU and a final max pooling that further down samples the feature maps. After these convolutional blocks, the feature maps are now reduced to a spatial dimension of 26x26 and then flattened to a single long vector. The flattened input is then fed into a fully connected dense layer that reduces the vector to 128 units, where another ReLU activation is applied to maintain non-linearity. A dropout layer follows, controlled by a tunable dropout rate parameter serving as regularization to prevent overfitting. Finally, a second layer maps the output to 3 units corresponding to the number of classes providing the raw logics for classification. All the tunable model parameters are optimized iteratively through the package, Optuna. It is able to search through a hyperparameter space to try parameters with increasing results with 30 trials. The training module then saves the CNN model with metadata so that it can be accessed later.

In the evaluation program, the process starts with importing the validation dataset that was saved in the data folder through the creation of a custom dataset class those images are prepared for processing. The evaluation function loads the train CNN checkpoint which includes saved meta data with the best hyperparameters. After setting the model to evaluation mode the program loads use a Dataloader method to iterate over the test samples in batches, for each batch the model computes output logits that are passed through a SoftMax function to obtain class probabilities with the predictions of true labels being collected to compute metrics. Included in those metrics are the model accuracy, precision, recall, and F1 score, while generating visualizations such a confusion matrix, ROC curve for each class, precision recall curves, and a summary bar plot of the metrics All of these, along with a detailed model information summary, are saved to a folder in the results directory.

## XGBoost

In the XGBoost training program, the process begins by setting random seeds for reproducibility and loading the training data, where the images are resized and flattened to form feature vectors appropriate for the model. The program then splits the data into training and validation subsets and uses Optuna to conduct hyperparameter tuning. During each trial of the objective function, the model is trained with a set of hyperparameters—such as max\_depth, eta, subsample, and colsample\_bytree—and its performance is evaluated on the validation set using early stopping. A custom progress bar callback (TqdmCallback) provides real-time feedback during the boosting rounds. Once the optimal set of hyperparameters is identified based on validation accuracy, the final model is re-trained using the best parameters, and its performance is further confirmed on the validation set. The best-performing model, along with its meta-data (including training parameters, best boosting iteration, and validation accuracy), is then saved to disk, enabling straightforward reloading for future evaluation or deployment. This model is trained using the system CPU due to issues with how it interacted with the local graphics card which is what led to the greatly reduced image size for training (32x32). These are in turn constructed through averaging the surrounding pixels.

The XGBoost evaluation program starts by loading the test dataset from preprocessed .npy files, where each image is resized (32×32 pixels) and then flattened into a one-dimensional array suitable for the XGBoost classifier. The program then loads the pre-trained XGBoost model along with its stored meta-data, which includes critical information such as the best boosting iteration and hyperparameters from the training phase. Using this information, the test data is wrapped into an XGBoost DMatrix and the model is used to predict class probabilities. The predicted probabilities are converted into class labels by taking the argmax across the output classes, after which standard metrics like accuracy, precision, recall, and F1 score are computed. In addition, several diagnostic plots—including a heatmap of the confusion matrix, ROC curves for each class, precision-recall curves, and a summary bar plot of the computed metrics—are generated and stored along with the evaluation metrics saved to a JSON file, ensuring comprehensive performance documentation.

## ViT Classifier

In the training process, the program initiates by setting seeds for reproducibility and configuring logging for real-time status updates. The training dataset is formed by loading images from .npy files and converting them into PIL images, which then undergo a series of randomized augmentations including resizing to 224×224 pixels, horizontal flips, and slight rotations to improve generalization. These transformed images are converted into tensors and normalized with standard ImageNet mean and standard deviation values. The SimpleViT model architecture is defined in the training script with a similar structure to the evaluation version: a patch embedding layer segments the input image into patches that are linearly projected into an embedding space; a class token is prepended to the patch embeddings, and fixed positional embeddings are added. This sequence is processed by a Transformer encoder consisting of a configurable number of layers (depth), multi-head self-attention with a tunable number of heads, and MLP blocks whose internal dimensions (mlp\_dim) are determined via hyperparameter tuning. The training employs Optuna to optimize various hyperparameters such as batch size, learning rate, embed\_dim, depth, heads, and mlp\_dim, ensuring that the embedding dimension is compatible with the number of heads. During each epoch, the model is trained using the Adam optimizer with gradient clipping and a StepLR scheduler to adjust the learning rate dynamically. After multiple epochs with periodic validation, the best checkpoint—based on validation accuracy—is saved along with a meta-data dictionary capturing critical model parameters (image\_size, patch\_size, num\_classes, heads, embed\_dim, depth, mlp\_dim) and the best hyperparameter values identified during tuning. This rigorous approach guarantees that the final model is not only well-tuned but also fully reproducible and ready for subsequent evaluation or deployment .

In the evaluation process, the program begins by loading the preprocessed test images and labels from .npy files. The images, initially normalized floats, are transformed back into 8-bit unsigned integers and converted into PIL images. A deterministic transformation pipeline is then applied, which includes resizing the images to 224×224 pixels, converting them into tensors, and normalizing them with ImageNet statistics. The evaluation code implements a custom dataset and DataLoader to iterate over the test samples in batches. The core of the model is a SimpleViT architecture, which starts with a convolutional patch embedding layer that divides the image into fixed-size patches while projecting them into an embedding space. This is followed by the inclusion of a learnable class token and a positional embedding that encodes the spatial relationships of the patches. The sequence of patch embeddings is then processed by a Transformer encoder comprising multiple layers—each having multi-head self-attention (with a defined number of heads), feedforward MLP components (with a configurable mlp\_dim), and residual connections—thereby capturing long-range dependencies in the image. Finally, the output corresponding to the class token is fed through a linear layer (mlp\_head) that maps the representation to the number of classes. The evaluation routine then computes softmax probabilities, aggregates predictions, and calculates comprehensive metrics such as accuracy, precision, recall, F1 score, and produces a series of diagnostic plots (e.g., confusion matrix, ROC curves, and precision-recall curves) to assess model performance, saving these results for further analysis.

## Masked CNN

The training workflow begins by constructing a binary mask that guides the CNN to focus only on the most relevant regions of each image. This is accomplished by combining interpretability techniques—LIME and RandomForest. LIME is applied to a small sample of training images using an untrained CNN, generating image-specific explanations that highlight the most influential superpixels for prediction. Each mask is saved visually and numerically. Simultaneously, a RandomForest classifier is trained on flattened pixel data from the training set and computes a global feature importance map, reshaped back to image space and averaged across color channels. Both the per-image LIME masks and the single RF importance map are normalized, then blended using a weighted average. The final result is a combined importance map, which is thresholded to form a binary mask that highlights areas deemed consistently significant by both explainers. This mask is then applied to the entire training, validation, and test datasets by zeroing out all unimportant pixels—effectively filtering the input data to emphasize semantically meaningful regions and reduce noise.

The CNN architecture is a three-stage convolutional network with ReLU activations and max pooling, followed by a fully connected head. However, before training begins, the architecture and its training parameters are tuned using Optuna, a powerful hyperparameter optimization framework. In the training script, Optuna defines a trial function where key parameters—such as learning rate, number of filters, number of layers, and possibly dropout or batch size—are treated as variables to explore. For each trial, a model is instantiated with the suggested configuration, trained on a subset of the masked data, and evaluated on a validation split. The evaluation metric (typically accuracy) is reported back to Optuna, which uses Bayesian optimization to guide the search. This cycle repeats 30 trials. Afterward, the best hyperparameter set is used to retrain the model on the full modified dataset (train and validation), and the best model checkpoint is saved. The script also stores associated meta-data, including the chosen architecture parameters, training hyperparameters, and paths to the masking files—ensuring reproducibility in both model structure and data preprocessing.

The evaluation script mirrors the training setup but applies it to unseen test data. First, it loads the normalized test images and converts any one-hot encoded labels to class indices. It retrieves the saved binary mask from disk and applies it to all test images, ensuring the same input filtering used during training. The data is wrapped into a PyTorch dataset with image resizing and normalization transforms, and a DataLoader is created for batching. The model is then rebuilt using parameters from the saved meta-data and loaded with the best-trained weights. Inference is performed on the masked test data in evaluation mode using softmax probabilities. Predictions and probabilities are collected, and standard metrics—accuracy, precision, recall, and F1 score—are calculated and printed. Additionally, the script generates and saves visualizations: a confusion matrix heatmap, ROC curves for each class, and precision-recall curves. A summary bar chart consolidates performance metrics, and all results are serialized into a JSON file for archival and programmatic access. This tightly integrated pipeline ensures that model evaluation is both transparent and faithful to the training setup, especially in how the binary mask shapes the model’s interpretation of its inputs.

# Dataset and Visualization

## Dataset Summary and Statistics

The dataset once processed contains X\_train and X\_test arrays containing images, and corresponding label arrays y\_train and y\_test. Each imaged is reduced down to a matrix of of uniform shape of 224x224x3 resulting in 150,528 features per image (224 × 224 × 3). In total there are 2266 images with the training set containing 1815 and the validation set with 451 images. The full split of each class by set is summarized in Table 1 below. This high-dimensional feature space is typical for raw image data.

Table 1: Count by class of images of meat

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Fresh | Half-Fresh | Spoiled |
| Train | 675 | 630 | 510 |
| Test | 178 | 159 | 114 |

Tests show that the dataset is complete without obvious typos or errors. All image files were matched to a label in the excel spreadsheet, no labels were saved without images as a result of running the matching code or vice versa. Leading to the conclusion that the dataset is complete and ready for model training. The significant number of features increases the model training complexity, however, that would be addressed later by including a masked CNN model for consideration amending the original project plan.

## Dimensionality Reduction with PCA and Visual Trends

To gain insight into how distinguishable the classes are in feature space, Principal Component Analysis (PCA) was applied. This technique projects high-dimensional image vectors down to two or three dimensions while retaining as much variance as possible. The 2D PCA scatter plot showed some separation between the three classes, particularly between Fresh and Spoiled categories, although Half-Fresh samples overlap with both, reflecting their intermediate characteristics (Figure 1).

This overlap introduces a novel challenge. While extreme categories (Fresh vs Spoiled) are visually and statistically separable, Half-Fresh examples represent a transition zone. This overlap may contribute to model confusion and motivate future work in either sharpening category boundaries or using probabilistic labels. For the purposes of the project, half-fresh meat is considered safe to eat but not fresh. This plot introduces some confusion into that regard. If there is a visual difference between fresh and partially fresh meat at what point does the author of the dataset conclude that the meat has spoiled. This seemingly arbitrary category description that was not elaborated on in the dataset readme file will complicate the accuracy of the resulting models and influence any interpretation of the results.

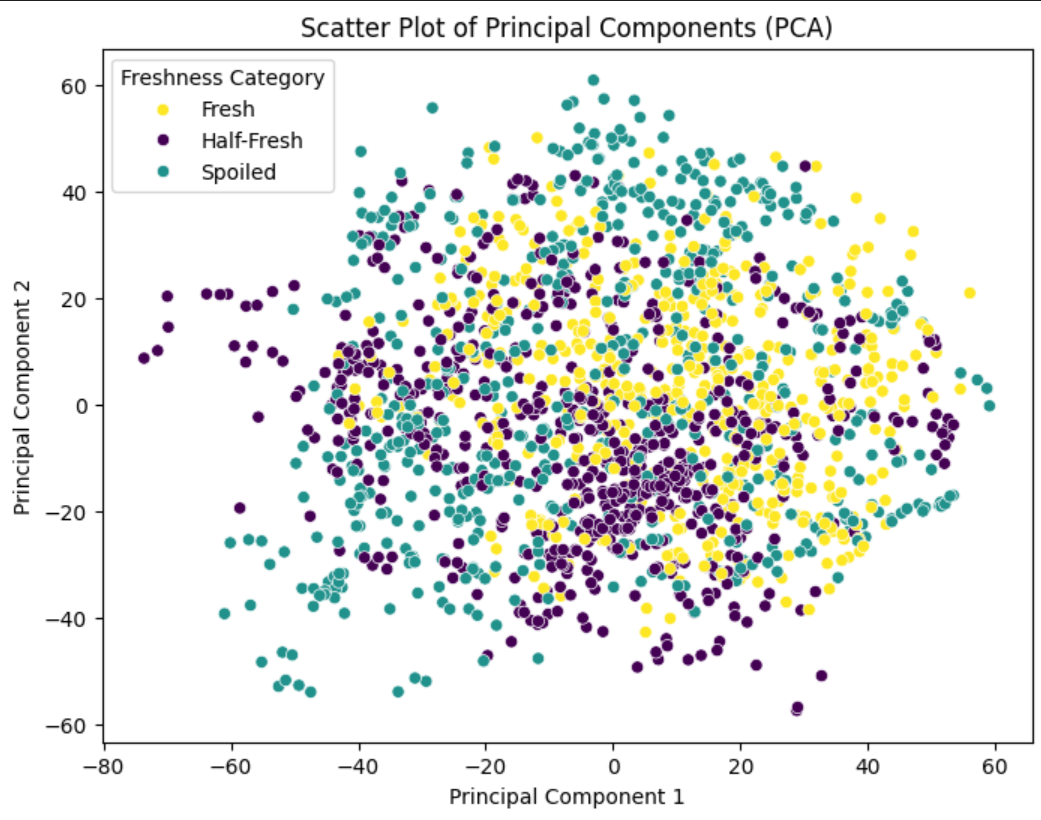


Figure 1: PCA Plot of Principal Components

## Pixel Intensity Histograms and Per-Class Visualization

Pixel value histograms were created for each class label. Fresh images exhibited a full range of brightness values with a strong peak around high pixel intensities (~200–255), possibly due to reflective surface characteristics of fresh meat (Figure 2). Half-Fresh images showed slightly lower peaks, indicating reduced contrast and light absorption consistent with early spoilage (Figure 3). In contrast, Spoiled images demonstrated a clear shift toward the lower end of the intensity spectrum with a prominent peak in the 0–50 range, highlighting areas of discoloration and decomposition (Figure 4). Most likely the spoilage images are biased by the presence of duller red color and brown. Limited specs of it are not significant when observing the half-fresh image histogram even if the overall color has declined. This per-class contrast in brightness distributions affirms the hypothesis that color and luminance features are meaningful indicators of freshness. These observations serve to further challenge the categorization of meat into fresh, half fresh and spoiled categories. The labeler probably used other methodologies to determine freshness of each meat which guide this determination, doing it solely visually may exclude important factors. In particular, the category of half-fresh is expected to cause challenges.

A graph of a person with a white background

AI-generated content may be incorrect.

Figure 2: Overlayed pixel value histogram for fresh images

A graph of a line graph

AI-generated content may be incorrect.

Figure 3: Overlayed pixel value histogram for half-fresh images

A graph of a number of images

AI-generated content may be incorrect.

Figure 4: Overlayed pixel value histogram for spoiled images

# Results and Discussion

## Individual Model Breakdown

### Simple CNN

The CNN created had the structure as defined above. This structure is recapped by Table 3 with the tuned hyper parameters as a result of Optuna running 30 trials in Table 2. When evaluated on the testing set the model performed well. The model metrics in table 4 are all at 90% besides the risk of misclassification risk for a meat that is spoiled being labeled as safe to eat with a 11.40% risk. The confusion matrix also supports that conclusion (Figure 5). For Fresh (Class 0), out of 178 samples, 168 were correctly classified, with 9 misclassified as Half-Fresh and 1 as Spoiled. Half-Fresh (Class 1) exhibits the greatest ambiguity: 143 correct classifications out of 159, with 15 false positives from Fresh and 1 from Spoiled. Spoiled (Class 2) had 101 correct out of 114, with 2 false positives from Fresh and 11 from Half-Fresh. The pattern here reveals that Half-Fresh overlaps visually with both adjacent categories, aligning with what’s seen in PCA plots from the EDA and confirmed by relatively lower precision and recall for that class in Table 5. Specifically, Half-Fresh has a precision of 0.88 and recall of 0.90, compared to Fresh at 0.91/0.94, and Spoiled at 0.98/0.89.

From a threshold-based perspective, the model’s ROC curves show that all classes are highly separable in probabilistic output space (Figure 7). Spoiled achieves the highest AUC at 0.997, Fresh follows at 0.988, and Half-Fresh at 0.971—still very high, though marginally lower. This confirms the model’s confidence in binary separations even when the categorical boundary is blurred. The precision-recall curves reflect the same: Spoiled maintains high precision even at high recall, while Half-Fresh dips substantially, with visible instability (jaggedness in the curve), indicating inconsistent predictions when recall increases (Figure 6). This instability corresponds with misclassification seen in the confusion matrix and underscores the need for better feature distinction between transitional samples (Figure 5). The tightly group set of Table 4 are further evidence of strong and consistent generalization. These results demonstrate that while the CNN confidently distinguishes Fresh and Spoiled samples, Half-Fresh remains a source of classification uncertainty, likely due to inherent overlap in visual features. Effects such as masking could lead to improvements in this area by limiting noise from camera angle variations and other factors such as the amount of image taken up by the meat.

Table 2: Simple CNN Model Information

|  |  |
| --- | --- |
| Total Parameters | 11,227,331 |
| Trainable Parameters | 11,227,331 |
| Number of Layers | 14 (incl. activations, dropout) |
| Output Classes | 3 |
| Dropout Rate | 0.3179 |
| Epochs | 20 |
| Optimizer | Adam |
| Batch Size | 32 |

Table 3: Simple CNN Layers with Tuned Best Hyperparameters

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer #** | **Type** | **Configuration Details** | **Output Shape\*** |
| 0 | Conv2d | in\_channels=3, out\_channels=128, kernel\_size=3×3 | (128, H-2, W-2) |
| 1 | ReLU | — | (128, H-2, W-2) |
| 2 | MaxPool2d | kernel=2×2, stride=2 | (128, H/2-1, W/2-1) |
| 3 | Conv2d | in\_channels=128, out\_channels=64, kernel\_size=3×3 | (64, H/2-3, W/2-3) |
| 4 | ReLU | — | (64, H/2-3, W/2-3) |
| 5 | MaxPool2d | kernel=2×2, stride=2 | (64, H/4-1.5, W/4-1.5) |
| 6 | Conv2d | in\_channels=64, out\_channels=128, kernel\_size=3×3 | (128, H/4-3.5, W/4-3.5) |
| 7 | ReLU | — | (128, H/4-3.5, W/4-3.5) |
| 8 | MaxPool2d | kernel=2×2, stride=2 | (128, ~26, ~26)\*\* |
| 9 | Flatten | — | (86528,) |
| 10 | Linear | in\_features=86,528, out\_features=128 | (128,) |
| 11 | ReLU | — | (128,) |
| 12 | Dropout | p=0.3179 | (128,) |
| 13 | Linear | in\_features=128, out\_features=3 | (3,) |

Table 4: Simple CNN Performance Metrics

|  |  |
| --- | --- |
| Performance Metric | Value |
| Accuracy | 0.91352 |
| Precision | 0.915566 |
| Recall | 0.913525 |
| F1 Score | 0.913756 |
| Model Size (MB) | 42.82 |
| Risk of Harming Consumer | 11.40% |

Table 5: Simple CNN Classification Report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Precision | Recall | F1-Score | Support |
| 0 (Fresh) | 0.91 | 0.94 | 0.93 | 178 |
| 1 (Half-Fresh) | 0.88 | 0.9 | 0.89 | 159 |
| 2 (Spoiled) | 0.98 | 0.89 | 0.93 | 114 |
| Overall Metrics |  |  |  | 451 |
| Accuracy |  |  | 0.91 |  |
| Macro Avg | 0.92 | 0.91 | 0.91 |  |
| Weighted Avg | 0.92 | 0.91 | 0.91 |  |

A blue squares with numbers and a number on it

AI-generated content may be incorrect.

Figure 5: Simple CNN Confusion Matrix

A graph of a line graph

AI-generated content may be incorrect.

Figure 6: Simple CNN Precision Recall Curve by Class

A graph of a curve

AI-generated content may be incorrect.

Figure 7: Simple CNN ROC Curves by Class

### XGBoost Classifier

The XGBoost classifier, trained on resized 32×32 meat images using pixel intensities as tabular features, achieved strong results with an overall accuracy of 82.9%, precision of 83.3%, and F1-score of 82.9% across the three freshness categories (Table 8). The model was optimized using Optuna, and the best hyperparameters selected were a maximum depth of 5, learning rate (eta) of 0.1886, subsample ratio of 0.627, and column sample by tree ratio of 0.5011, over 98 boosting rounds (Table 6). Shallow trees with limited feature exposure per iteration help avoid overfitting on small high-dimensional image vectors. Evaluation reveals that the model performs consistently across all classes, with slight differences in confusion rates. The confusion matrix shows that Class 0 (Fresh) had 149 true positives, misclassified 20 times as Half-Fresh and 9 times as Spoiled. Class 1 (Half-Fresh), often difficult to distinguish, had 138 correct predictions, with only 15 and 6 samples misclassified into Fresh and Spoiled, respectively. Class 2 (Spoiled) had 87 correct classifications, with its errors mostly going to Half-Fresh (20 samples) (Figure8). The distribution indicates XGBoost’s relative success at minimizing confusion across class boundaries, especially compared to the CNN. These numbers translate into per-class performance scores: Fresh (Class 0) reached 0.87 precision and 0.84 recall, Half-Fresh (Class 1) had 0.78 precision and 0.87 recall, while Spoiled (Class 2) achieved 0.85 precision and 0.76 recall the classification report (Table 7). The ROC curve further confirms that the classifier maintains high separability across all classes, with AUC values of 0.97 (Class 0), 0.96 (Class 1), and 0.97 (Class 2) (Table 8, Figure 10). These high AUC scores indicate strong discriminative capacity at various classification thresholds and demonstrate that the softprob output probabilities are well calibrated. The precision-recall curves reveal minimal degradation in precision as recall increases, particularly for Classes 0 and 1, where the curves remain close to the top right of the plot—suggesting the model maintains confidence even while expanding recall (Figure 9). Class 2’s curve drops slightly sooner, aligning with its comparatively lower recall in the confusion matrix and indicating that spoiled samples may be harder to identify consistently using raw pixel intensities alone. Which is an observation that counter conventional logic with the spoiled samples typically being more readily distinguishable. The model’s overall architecture and training procedure shows that even a light XGBoost model after careful tuning can compete with its heavier counterparts. Despite not leveraging spatial features directly, this model benefits from tree-based nonlinearity and probabilistic prediction, showing that for structured and flattened visual inputs, XGBoost can produce stable and explainable outcomes which may be particularly attractive to manufacturing enterprises.

Table 6: Optuna Results for XGBoost Tuning

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| Objective | multi:softprob |
| Number of Classes | 3 |
| Evaluation Metric | mlogloss |
| Seed | 42 |
| Resize Shape | 32 × 32 |
| Best Iteration | 98 |
| **Hyperparameter** | **Value** |
| Max Depth | 5 |
| Learning Rate (eta) | 0.1886 |
| Subsample Ratio | 0.6275 |
| Column Subsample | 0.5011 |
| **Optimized Result** | **Value** |
| Validation Accuracy | 0.9118 |

Table 7: XGBoost Classification Report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Precision | Recall | F1-Score | Support |
| 0 (Fresh) | 0.87 | 0.84 | 0.85 | 178 |
| 1 (Half-Fresh) | 0.78 | 0.87 | 0.82 | 159 |
| 2 (Spoiled) | 0.85 | 0.76 | 0.81 | 114 |
| Overall Metrics |  |  |  | 451 |
| Accuracy |  |  | 0.83 |  |
| Macro Avg | 0.83 | 0.82 | 0.83 |  |
| Weighted Avg | 0.83 | 0.83 | 0.83 |  |

Table 8: XGBoost Model Metrics

|  |  |
| --- | --- |
| Metric | Value |
| Accuracy | 0.8293 |
| Precision | 0.8328 |
| Recall | 0.8293 |
| F1-Score | 0.8294 |
| AUC Score Fresh | 0.9698 |
| AUC Score Half-Fresh | 0.9648 |
| AUC Score Spoiled | 0.9683 |
| Model Size (MB) | 0.65 |
| Risk of Harming Consumer | 23.68% |

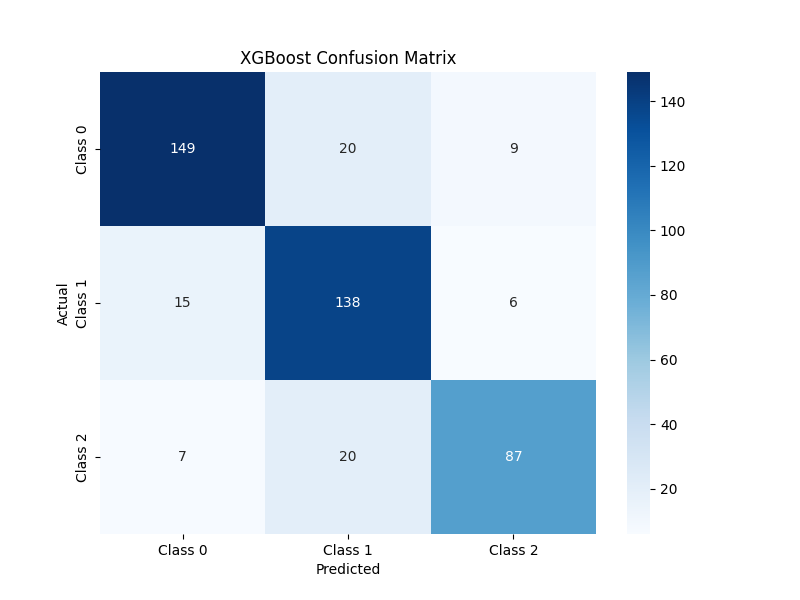


Figure 8: XGBoost Confusion Matrix

A graph of a curve

AI-generated content may be incorrect.

Figure 9: XGBoost Precision Recall Curve by Class

A graph of a curve

AI-generated content may be incorrect.

Figure 10: XBGoost ROC Curve by Class

### ViT Model

The Vision Transformer (ViT) model implemented uses a simplified transformer-based architecture designed to process 224×224 images through a patch embedding strategy. The model begins by converting each image into 16×16 patches using a convolutional layer, which maps the 3-channel RGB image to an embedding space of dimension 768, one of the hyperparameters optimized through Optuna. A learnable class token is prepended to the patch sequence and added to a positional encoding before being fed into a Transformer encoder with 8 layers (depth=8). Each encoder layer contains 12 attention heads and a feedforward block with dimensionality 2048, giving the model sufficient capacity to learn both spatial relationships and global context across the image. After encoding, the output corresponding to the class token is passed through a linear MLP head to produce logits for 3 freshness classes. This configuration—heads=12, embed\_dim=768, depth=8, mlp\_dim=2048—was selected through Optuna during training.

The evaluation results show that the ViT model achieves an overall accuracy of 83.4%, precision of 84.8%, and an F1-score of 82.8%, slightly outperforming the XGBoost baseline and aligning closely with the standard CNN in some metrics(Table 10). Class-level performance, detailed in the confusion matrix, reveals that the model handles Fresh and Spoiled samples extremely well. Class 0 (Fresh) had 170 correct predictions out of 178 (recall=0.96), with just 6 misclassified as Half-Fresh and 2 as Spoiled. Class 2 (Spoiled) had 107 correct out of 114 (recall=0.94), with almost all confusion happening with Half-Fresh. Class 1 (Half-Fresh), as expected, presents the most difficulty. The model correctly predicted 99 out of 159, with 56 samples misclassified as Fresh—suggesting an over-reliance on certain low-level visual features possibly shared across these categories. Nevertheless, Half-Fresh still maintains high precision (0.88) due to relatively few false positives from other classes. The ROC curves illustrate excellent separability across all classes, with AUC values of 0.944 (Class 0), 0.928 (Class 1), and an exceptional 0.997 (Class 2) (Table 10, Figure 13). These numbers confirm that the transformer is able to form high-confidence boundaries between classes even under probabilistic prediction thresholds. The PR curves provide a more detailed view of the model’s trade-offs (Figure 12). Class 2’s curve is the most stable, maintaining high precision across all recall levels. Class 0 shows similar instability to Class 1, with a noticeable drop-off in precision at mid-to-high recall. Revealing an interesting observation, that half-fresh meat is closer from the models perspective to Fresh meat than to spoiled meat. These results validate the ViT’s strength in capturing global texture and structural cues, particularly useful for identifying visually distinct categories like Spoiled meat. However, the heavy overlap between Fresh and Half-Fresh samples continues to challenge even advanced architectures. The model benefits from Optuna’s tuning across embedding size, head count, depth, and MLP width—each layer contributes to a hierarchical understanding of spatial patterns in the image. The ViT shows outperforms the other models when it comes to segmenting fresh and spoiled meats with a size comparable to either CNN. These factors would garner this classification further attention when choosing a strategy to implement.

Table 9: ViT Classification Report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Precision | Recall | F1-Score | Support |
| 0 (Fresh) | 0.75 | 0.96 | 0.84 | 178 |
| 1 (Half-Fresh) | 0.88 | 0.62 | 0.73 | 159 |
| 2 (Spoiled) | 0.95 | 0.94 | 0.94 | 114 |
| Overall Metrics |  |  |  | 451 |
| Accuracy |  |  | 0.83 |  |
| Macro Avg | 0.86 | 0.84 | 0.84 |  |
| Weighted Avg | 0.85 | 0.83 | 0.83 |  |

Table 10: ViT Model Performance Metrics

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Accuracy | 0.8337 |
| Precision (Weighted Avg) | 0.8479 |
| Recall (Weighted Avg) | 0.8337 |
| F1 Score (Weighted Avg) | 0.828 |
| ROC AUC – Class 0 (Fresh) | 0.944 |
| ROC AUC – Class 1 (Half-Fresh) | 0.9281 |
| ROC AUC – Class 2 (Spoiled) | 0.9971 |
| Model Size (MB) | 38.4 |
| Input Resolution | 224 × 224 |
| Patch Size | 16 × 16 |
| Embedding Dimension | 768 |
| Number of Transformer Layers (Depth) | 8 |
| Number of Attention Heads | 12 |
| MLP Hidden Dimension | 2048 |
| Risk to Consumer | 6.14% |

A blue squares with white text

AI-generated content may be incorrect.

Figure 11: ViT Confusion Matrix

A graph of a curve

AI-generated content may be incorrect.

Figure 12: ViT Precision-Recall Curves per Class

A graph of a curve

AI-generated content may be incorrect.

Figure 13: ViT ROC Curves Per Class

### Masked CNN

The Modified CNN model uses a unique preprocessing step driven by model interpretability. Instead of training directly on raw images, this pipeline constructs a binary spatial mask by aggregating saliency maps generated via LIME and pixel-level feature importances estimated with a Random Forest classifier. The top 10 LIME masks are averaged and normalized, as is the RF feature importance map, and the two are then blended at a 0.5 weighting to create a combined relevance score. This 4 step process is captured in Figure A threshold of 0.5 is applied to produce a binary mask which is broadcast across the RGB channels and element-wise multiplied with each image. This transformation removes irrelevant or low-importance regions before the model sees the data, effectively filtering the dataset spatially based on interpretability-driven logic. The CNN model structure itself is unchanged from the baseline: it consists of three convolutional layers with 32, 64, and 128 filters respectively (all with 3×3 kernels and ReLU activations), followed by 2×2 max pooling operations. The classifier head flattens the resulting feature map (28×28×128 = 100,352 units), feeds it through a fully connected layer with 128 hidden units and another ReLU activation, then outputs logits to the final 3-way classification layer. All training was done with the Adam optimizer, a learning rate of 0.001, and a batch size of 32 over 20 epochs.

Despite the promising motivation behind the mask, the model’s test set performance degrades notably achieving a 60.1% accuracy, with a precision of 59.7%, recall of 60.1%, and F1 score of 59.4% (Table 12). The per-class breakdown shows Class 2 (Spoiled) had the highest recall at 0.75 and F1-score at 0.67, while Class 1 (Half-Fresh) underperformed with a recall of 0.43 and F1 of 0.49, reinforcing concerns that the aggressive spatial masking suppresses subtle texture patterns critical for distinguishing transitional states (Table 11). The confusion matrix reveals the model is still strongest on Class 2 (Spoiled), correctly classifying 86 of 114 images and achieving a recall of 0.75, though with slightly more balanced misclassifications between Fresh and Half-Fresh. Fresh samples (Class 0) were predicted correctly 116 times but saw 40 misclassified as Half-Fresh, and 22 as Spoiled (Figure 15). Half-Fresh (Class 1) had only 69 correct classifications out of 159—a recall of just 0.43—and was heavily confused with both other categories, especially Fresh. This is consistent with the model having lost subtle boundary information due to excessive masking. The ROC curves reflect the classifier's uncertainty, with AUC scores of 0.767 (Fresh), 0.699 (Half-Fresh), and 0.869 (Spoiled). Compared to other models, the ROC for Half-Fresh dips below 0.70, indicating weak confidence in its probability outputs for this class (Figure 17). The [precision-recall curves] show that while Spoiled maintains stable performance across thresholds, Half-Fresh again drops sharply as recall increases, and Fresh starts strong but degrades (Figure 16). The masking introduced a blunt filtering effect that removed too much class relevant content. Reducing the strength of the filter in the future may help the filter with accomplishing its task of eliminating the image background. On the other hand, as it stands it still notes important areas of meat that tend to spoil first which may be a valuable observation when it comes to evaluating the process. This data may be worth storing regardless of the model type being used. Adaptive maxing driving by tuning would be a could optimize the model perhaps beyond the superb performance of the simple CNN.

Table 11: Masked CNN Classification Report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Precision | Recall | F1-Score | Support |
| 0 (Fresh) | 0.63 | 0.65 | 0.64 | 178 |
| 1 (Half-Fresh) | 0.56 | 0.43 | 0.49 | 159 |
| 2 (Spoiled) | 0.6 | 0.75 | 0.67 | 114 |
| Overall Metrics |  |  |  | 451 |
| Accuracy |  |  | 0.6 |  |
| Macro Avg | 0.6 | 0.61 | 0.6 |  |
| Weighted Avg | 0.6 | 0.6 | 0.59 |  |

Table 12: Masked CNN Performance Summary

|  |  |
| --- | --- |
| Metric | Value |
| Accuracy | 0.6009 |
| Precision (Weighted Avg) | 0.5973 |
| Recall (Weighted Avg) | 0.6009 |
| F1 Score (Weighted Avg) | 0.5939 |
| ROC AUC – Class 0 (Fresh) | 0.7672 |
| ROC AUC – Class 1 (Half-Fresh) | 0.6992 |
| ROC AUC – Class 2 (Spoiled) | 0.8694 |
| Model Size (MB) | 42.83 |
| Risk to Consumer | 24.56% |

A screenshot of a screen capture

AI-generated content may be incorrect.

Figure 14: Feature Screening Masks For Masked CNN

A blue squares with numbers and a number

AI-generated content may be incorrect.

Figure 15: Masked CNN Confusion Matrix

A graph of different colored lines

AI-generated content may be incorrect.

Figure 16: Masked CNN Precision-recall Curves per Class

A graph of a number of different colored lines

AI-generated content may be incorrect.

Figure 17: ROC Curves Per Class

## Comparative Results

Among the four evaluated models, the Simple CNN demonstrated the best overall performance, achieving the highest accuracy (91.4%) and F1-score, while maintaining a manageable size (~44 MB), making it ideal for deployment in quality control systems on the factory floor. The Vision Transformer (ViT) closely matched CNN performance (83.4% accuracy) but with significantly higher model complexity (~38.4 MB), limiting its practicality in real-time or resource-constrained environments. XGBoost emerged as a lightweight (0.65 MB), high-speed alternative with strong accuracy (83.3%) and excellent AUC values, particularly suitable for embedded systems that rely on structured pixel data. In contrast, the Modified CNN, which used a binary mask derived from LIME and Random Forest interpretability methods, performed poorly (60.1% accuracy), likely due to critical feature suppression during masking. Though conceptually interpretable, this approach eliminated spatial context essential for accurate classification. A comparison of the 4 approaches is given below in Table 13. Additionally, exploratory K-Means clustering performed in the EDA confirmed that unsupervised methods fail to capture the complexity of meat freshness without supervised learning—further reinforcing the need for end-to-end models that learn visual structure directly from labeled data. Consequently, the following approach seems most favorable: developing a tunable masking strategy and then implementing it for the ViT rather than the CNN. By being more effective at identifying the distinction between spoiled and fresh meat the ViT model reduced risks to consumers while offering similar size and a comparable accuracy to the other approaches. For concerns, more challenged in terms of computational prowess it would be prudent to pursue the XGBoost classifier.

Table 13: Comparison of Model Performance Across All Classifiers

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1 Score | Avg AUC | Params | Size (MB) | Risk to Consumers |
| Simple CNN | 0.914 | 0.916 | 0.914 | 0.914 | ~0.985 | 11.2M | 42.83 | 11.40% |
| ViT | 0.834 | 0.848 | 0.834 | 0.828 | 0.956 | ~22.3M | 38.04 | 6.14% |
| XGBoost | 0.833 | 0.833 | 0.833 | 0.829 | 0.967 | ~0.1M | 0.65 | 23.68% |
| Modified CNN | 0.601 | 0.597 | 0.601 | 0.594 | 0.779 | 11.2M | 42.83 | 24.56% |

# Conclusion

This project confirms that machine learning can reliably automate meat freshness classification, reducing reliance on human inspection. The simple CNN achieved the highest accuracy (91.4%) but at a larger model size, while the ViT offered balanced performance (83.4% accuracy, 6.1% consumer risk) with global attention benefits. XGBoost proved an efficient alternative for low‑resource settings, and the masked CNN highlighted the challenges of interpretability‑driven preprocessing. Model selection should be driven by deployment constraints: for high‑throughput lines with adequate hardware, the CNN or ViT is recommended; for embedded or edge devices, XGBoost offers a compact solution. Future work should refine masking thresholds and explore hybrid ViT‑masking strategies to further minimize misclassification of spoiled meat. Efforts should also be taken to define and understand the category of half-fresh to perhaps link other non-visuals factors such as duration of exposure to open air.

# Data and Code

Data obtained from: <https://www.kaggle.com/datasets/vinayakshanawad/meat-freshness-image-dataset>

Full project code at: <https://github.com/sfeldmanMIG25/Meat_Labeler>

# References

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM. https://doi.org/10.1145/2939672.2939785

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., … Houlsby, N. (2021). An image is worth 16×16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. <https://doi.org/10.48550/arXiv.2010.11929>

Elangovan, P., Dhurairajan, V., Nath, M. K., Yogarajah, P., & Condell, J. (2024). A novel approach for meat quality assessment using an ensemble of compact convolutional neural networks. *Applied Sciences, 14*(14), 5979. <https://doi.org/10.3390/app14145979>

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). “Why should I trust you?”: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135–1144). ACM. https://doi.org/10.1145/2939672.2939778

Song, S., Guo, Q., Duan, X., Shi, X., & Liu, Z. (2024). Research on pork cut and freshness determination method based on computer vision. *Foods, 13*(24), 3986. <https://doi.org/10.3390/foods13243986>

Taheri-Garavand, A., Fatahi, S., Shahbazi, F., & de la Guardia, M. (2019). A nondestructive intelligent approach to real-time evaluation of chicken meat freshness based on computer vision technique. *Journal of Food Process Engineering, 42*, e13039. <https://doi.org/10.1111/jfpe.13039>