ENPM703 Fundamentals of AI and Deep Learning

Project Architecture Report

Phishing Email Detection with Multimodal Deep Learning

Version #:	1.0
Version Date:	12/18/2025
Team Name:	Group 3
Contributing Author(s):	Vishal Patil, Srihari Narayan, Anila Sai Namburi, Akash Vora

Table of Contents

1 Introduction

This report documents an end-to-end phishing detection system built as a Multimodal Fusion Architecture.

The system combines three complementary signals:

(1) email text content,
(2) brand-logo imagery,
(3) engineered metadata features (such as URL and sender-domain properties).

Two specialist neural networks are first trained independently (a 1D-CNN for text and a 2D-CNN for logos). A final fusion model then integrates their embeddings with metadata to produce a phishing probability for real HTML emails.

1.1 Key Outcomes

- Higher robustness than text-only detection by validating brand context (e.g., PayPal mentioned in text but logo mismatch).
- Modular training via specialist pre-training followed by fusion training.
- Production artifact: best_fusion_model.pth, used by a preprocessing-and-prediction script for HTML emails.

Metric	Value
Text CNN Accuracy	98.96%
Image CNN Accuracy	76.30%
Fusion Model Accuracy	99.45%
ROC-AUC Score	0.9998

2 System Goals and Threat Model

The system is designed to classify incoming emails as phishing or legitimate by combining textual cues like social engineering, urgency, credential requests and visual cues like brand impersonation through logos, and structural cues like URLs, sender domain patterns, message composition.

2.1 In-scope attack patterns

- Credential-harvesting emails with urgency and call-to-action links.
- Brand impersonation where the email claims to be from a known organization and includes an embedded logo.
- HTML emails using obfuscated text, excessive links, shortened URLs, or mismatched sender domains.

2.2 Out-of-scope

- Highly targeted spear-phishing with minimal text and no logos.
- Attacks that rely entirely on attachments or QR codes unless added as modalities.
- Non-email channels like SMS/WhatsApp unless the same preprocessing and data are adapted.

3 Architecture Overview

3.1 System Design Philosophy

The multimodal phishing detection system is built on a four-phase architecture that separates concerns, enables modular training, and supports production deployment. The design follows three core principles:

Principle 1: Specialize Then Fuse

Rather than training a single end-to-end model from scratch, the system first trains specialist neural networks for each modality independently. The text specialist learns to detect phishing language patterns, urgency cues, and credential requests. The image specialist learns to recognize brand logos and detect visual impersonation. These pretrained specialists later serve as feature extractors in the fusion architecture, providing strong initialization and enabling efficient joint training.

Principle 2: Explicit Data Linking

Phishing detection requires understanding the relationship between what an email claims (text) and what it shows. The architecture explicitly links text brand mentions to corresponding logo images through a multimodal connector pipeline. This linking enables the fusion model to detect brand mismatches, such as an email claiming to be from PayPal but displaying a suspicious or mismatched logo.

Principle 3: Consistent Artifact Interfaces

The system enforces strict consistency between training and inference through versioned artifacts. The vocabulary mapping, logo class indices, and metadata feature engineering logic are identical during deployment.

3.2 High-Level Architecture Flow

The system processes emails through four distinct phases:

Phase 1: Specialist Pre-training

This phase trains two independent neural networks on separate datasets. The text CNN processes 76,346 cleaned emails to learn textual phishing patterns, producing a vocabulary of 207,037 tokens and trained weights achieving 98.96% validation accuracy. The image CNN processes 72,652 brand logos across 352 classes to learn visual brand characteristics, producing trained weights achieving 76.30% validation accuracy. Both specialists output fixed-dimensional embeddings that capture semantic meaning in their respective modalities.

Phase 2: Multimodal Data Integration

This phase creates the golden dataset by aligning text and image modalities. A multimodal connector scans email text for brand mentions using regex patterns, extracts 20 metadata features like URL statistics, urgency keywords, sender patterns, etc, and links detected brands to corresponding logo image IDs from the OpenLogo dataset. The output is a unified CSV (unified_multimodal_text.csv) containing 49,860 samples where each row includes email text, a

20-dimensional metadata vector, and one or more image IDs representing logos that should appear in that email context.

Phase 3: Fusion Model Training

This phase trains the final classifier that integrates all three information sources. A custom PyTorch Dataset loads text tokens, logo images, and metadata vectors synchronously. The fusion model architecture consists of two towers: the text tower extracts 256-dimensional text embeddings, the image tower extracts 512-dimensional logo embeddings, and a small MLP projects metadata to 64 dimensions. These three vectors are concatenated with 832 total dimensions and passed through a deep classifier with four fully-connected layers to produce binary phishing predictions. The final model achieves 99.45% validation accuracy.

Phase 4: Inference Pipeline

This phase deploys the trained model for real-world classification. A Python script accepts raw HTML email files and extracts all necessary features: text is parsed from HTML, tokenized using the training vocabulary, truncated to 512 tokens, and converted to numerical indices. Embedded or linked images are extracted and preprocessed to 224×224 pixels with ImageNet normalization. If no image is present, a zero-filled dummy tensor is used. Metadata features are computed using the same logic as training. The fusion model runs in evaluation mode to produce a phishing probability and confidence score. The output includes the predicted class, confidence percentage, and a detailed report highlighting suspicious signals such as shortened URLs, urgency language, domain mismatches, or logo inconsistencies.

3.3 Architectural Components and Data Flow

Component 1: Text Processing Tower

- Input: Raw email subject and body text
- Processing: Cleaning → Tokenization → Vocabulary mapping → Embedding → 1D convolutions
- Architecture: Embedding layer (vocab_size → 128 dims) → 4 Conv1D blocks with progressive channels (64→128→256→512) → Global average pooling → FC projection to 256 dims
- Output: 256-dimensional text feature vector capturing semantic content and linguistic patterns

Component 2: Image Processing Tower

- Input: Brand logo image (RGB, variable size)
- Processing: Resize to 224×224 → Normalize with ImageNet statistics → VGG-style convolutions
- Architecture: 4 Conv2D blocks with progressive channels (64→128→256→512) → Adaptive pooling → FC projection to 512 dims
- Output: 512-dimensional image feature vector capturing visual brand identity

Component 3: Metadata

- Input: 20-dimensional numerical feature vector
- Processing: Linear projection → ReLU activation → Dropout regularization
- Architecture: FC layer (20 → 64 dims) with ReLU and dropout (p=0.25)
- Output: 64-dimensional metadata feature vector capturing statistical and structural properties

Component 4: Fusion Classifier

- Input: Concatenated feature vector (256 text + 512 image + 64 metadata = 832 dims)
- Processing: Deep fully-connected network with batch normalization and dropout
- Architecture: FC 832→512 → BN → ReLU → Dropout → FC 512→256 → BN → ReLU → Dropout → FC 256→128 → BN → ReLU → Dropout → FC 128→2 → Softmax
- Output: Binary classification logits and probabilities (legitimate vs phishing)

3.4 Key Design Decisions and Rationale

Decision 1: Brand-Based Image Linking

Rather than randomly pairing emails with logos, the system explicitly detects brand mentions in email text and links them to corresponding logo images. This enables the model to learn meaningful cross-modal relationships, such as recognizing when an email claims to be from PayPal (text) but displays a mismatched logo (image). The brand detection covers financial institutions, tech companies, e-commerce platforms, and shipping providers commonly targeted in phishing attacks.

Decision 2: Metadata

In addition to text and images, the architecture incorporates engineered metadata features as a third modality. These 20 dimensions capture signals that pure text or image analysis might miss, including URL characteristics (shortened links, IP addresses, suspicious TLDs), sender patterns (domain mismatches), structural properties (HTML complexity), and stylistic cues (excessive punctuation, capitalization). The metadata encoder projects these features to a 64-dimensional learned representation that complements the semantic understanding from text and visual understanding from images.

Decision 3: Dummy Tensors for Missing Modalities

Not all emails contain brand logos, and the training data includes text-only samples. Rather than filtering these out, the system uses zero-filled dummy tensors for missing images during both training and inference. This allows the model to learn to rely more heavily on text and metadata features when image information is unavailable, improving robustness in real-world scenarios where not all phishing emails contain logos.

3.5 System Architecture Diagram

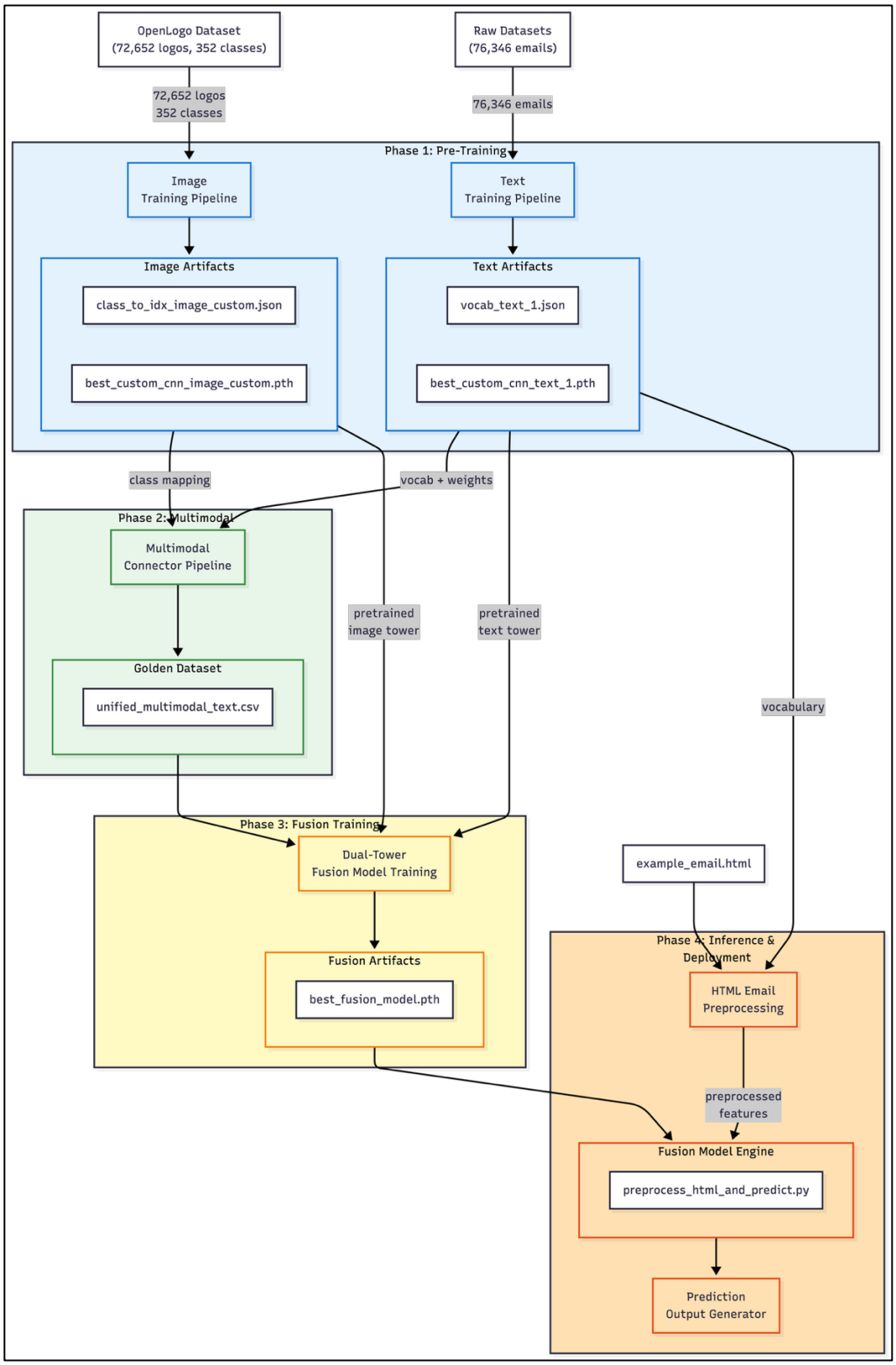

Fig: Multimodal Fusion System Architecture Diagram

4 Data, Artifacts, and Interfaces

The phishing detection pipeline produces and consumes multiple artifacts across four distinct phases: specialist pre-training, multimodal data integration, fusion model training, and inference. These artifacts must remain consistent across training and deployment to avoid train-serve skew and ensure reproducible results.

4.1 Raw Data Sources

The system processes two primary data sources that are combined during the multimodal integration phase.

4.1.1 Text Data Sources

- **CEAS_08 dataset:** Conference on Email and Anti-Spam 2008 challenge dataset containing labeled phishing and legitimate emails
- **Enron dataset:** Legitimate corporate email corpus
- **Nazario dataset:** Curated phishing email samples
- **Nigerian Fraud dataset:** Advance-fee fraud emails
- **SpamAssassin dataset:** Additional spam and phishing samples

Combined, these sources provide 76,346 email samples with fields including sender, receiver, date, subject, body, URLs, and binary labels (0=legitimate, 1=phishing).

4.1.2 Image Data Source

- **OpenLogo dataset:** 72,652 brand logo images across 352 classes representing major corporations, financial institutions, tech companies, and e-commerce platforms
- Images stored in JPEGImages directory with standardized naming convention
- Variable sizes and aspect ratios requiring preprocessing for model input

4.2 Phase 1 Artifacts: Pre-training

Phase 1 trains two independent specialist models that later serve as feature extractors in the fusion architecture.

4.2.1 Text Specialist Artifacts

- ***cleaned_combined_emails.csv:*** Merged and cleaned text dataset after standardization, deduplication, and noise removal (76,346 samples).
- ***best_custom_cnn_text_1.pth:*** Trained weights for the 1D-CNN text classifier achieving 98.96% validation accuracy (model size: approximately 15MB).
- ***vocab_text_1.json:*** Token-to-index mapping containing 207,037 unique vocabulary tokens with special tokens <pad>, <unk>, and index-to-token array.

The text CNN architecture consists of an embedding layer (vocab_size → 128 dimensions), four 1D convolutional blocks with progressive channel expansion (64→128→256→512), global average pooling, and fully connected projection to 256-dimensional feature space.

4.2.2 Image Specialist Artifacts

- ***best_custom_cnn_image_custom.pth:*** Trained weights for the VGG-style 2D-CNN logo classifier achieving 76.30% validation accuracy on 352 brand classes (model size: approximately 45MB).
- ***class_to_idx_image_custom.json:*** Brand name to class index mapping (352 entries) used to link text brand mentions to specific logo classes.

The image CNN follows a VGG-style architecture with four convolutional blocks (64→128→256→512 channels), adaptive pooling, and fully connected projection to 512-dimensional feature space.

4.3 Phase 2 Artifacts: Multimodal Data Integration

Phase 2 creates the golden dataset that enables joint training by linking email text with corresponding brand logos and computing metadata features.

4.3.1 Integration Pipeline Artifacts

- ***dual_tower_text_features.ipynb:*** This python notebook is used to integrate and generate the *unified_multimodal_text* dataset.
- ***unified_multimodal_text.csv:*** Golden dataset combining text, metadata, and image references (49,860 aligned samples after filtering)

4.3.2 Golden Dataset Schema

The unified CSV contains the following critical columns:

- **source:** Original dataset identifier (CEAS_08, Nazario, etc.)
- **sender, receiver, date:** Email metadata fields
- **subject, body:** Email text content
- **label:** Binary classification target (0=legitimate, 1=phishing)
- **sender_domain:** Extracted domain from sender email address
- **url_list:** Extracted URLs from email body
- **num_urls:** Count of URLs in email
- **url_domains:** Unique domains across all URLs
- **has_urgent_words:** Binary flag for urgency language detection
- **num_urgent_keywords:** Count of urgency-related terms
- **subject_length, body_length:** Character counts
- **sender_domain_suspicious:** Flag for suspicious sender patterns
- **domain_brand_mismatch:** Flag indicating sender domain differs from mentioned brand
- **receiver_domain, sender_receiver_same_domain:** Domain comparison flags
- **metadata_vector:** 20-dimensional numerical feature vector (stored as string representation of list)

- **mentioned_brands:** Detected brand names from text scanning
- **num_brands_mentioned:** Count of detected brands
- **has_brand_mention:** Binary flag indicating brand presence
- **image_brand_ids:** Comma-separated list of logo class indices matching mentioned brands
- **has_image_mappable_brand:** Binary flag indicating at least one brand has available logo in dataset

4.4 Phase 3 Artifacts: Fusion Model Training

Phase 3 trains the final multimodal classifier that integrates all three information sources.

4.4.1 Fusion Training Artifacts

- ***train_fusion3.ipynb:*** Training notebook implementing two-stage optimization (freeze-then-finetune)
- ***best_fusion_model.pth:*** Production-ready fusion model achieving 99.45% validation accuracy (model size: approximately 62MB)

4.4.2 Fusion Model Architecture

The Dual Tower Fusion Model integrates three feature extractors:
- **Text tower:** Pretrained Text Feature Extractor producing 256-dimensional embeddings
- **Image tower:** Pretrained Image Feature Extractor producing 512-dimensional embeddings
- **Metadata:** Small MLP projecting 20-dimensional metadata to 64 dimensions

The fusion classifier concatenates all three vectors (256+512+64=832 dimensions) and processes through four fully-connected layers (832→512→256→128→2) with batch normalization, ReLU activation, and dropout regularization.

4.4.3 Training Strategy

- **Epochs:** 10
- **Batch size:** 16 (limited by GPU memory with three modalities)
- **Optimizer:** Adam with default beta parameters
- **Loss:** Cross-entropy loss
- **Data split:** 80% training (39,888 samples), 20% validation (9,972 samples)

4.5 Phase 4 Artifacts: Inference Pipeline

Phase 4 deploys the trained model for real-world email classification.

4.5.1 Inference Artifacts

- **preprocess_html_and_predict.py:** Production inference script accepting raw HTML email files.
- **Model checkpoints:** best_fusion_model.pth, vocab_text_1.json.
- **Environment requirements:** PyTorch, PIL, BeautifulSoup, torchvision.

4.5.2 Inference Pipeline Steps

1. **HTML parsing:** Extract visible text, remove markup, detect hyperlinks, compute HTML statistics.
2. **Image extraction:** Capture embedded images (base64-encoded, external URLs, local files).
3. **Text preprocessing:** Tokenize using training vocabulary, truncate/pad to 512 tokens, convert to numerical indices.
4. **Image preprocessing:** Resize to 224×224 pixels, normalize using ImageNet statistics, create dummy tensor if no image available.
5. **Metadata extraction:** Compute same 20 features used during training, apply normalization.
6. **Model inference:** Run fusion model in eval mode, compute softmax probabilities.
7. **Output generation:** Return predicted class (legitimate/phishing), confidence score (0-100%), and detected issues report.

5 Phase 1: Specialist Pre-training

Phase 1 trains two independent specialists: a text classifier and a logo classifier. These specialists are later reused as feature extractors (towers) in the fusion model.

5.1 Text Specialist (1D-CNN)

Scripts: phish-email-detector-text.ipynb → Final_CNN_Text_1.ipynb.

- **Cleaning:** removes noisy characters and normalizes raw datasets; outputs cleaned_combined_emails.csv.
- **Tokenization:** builds a vocabulary and converts text to integer sequences using vocab_text_1.json.
- **Learning:** trains a 1D-CNN to detect phishing language patterns (urgency, credential prompts, spoofing terms).

Primary outputs: best_custom_cnn_text_1.pth (weights) and vocab_text_1.json (token mapping).

5.2 Image Specialist (2D-CNN, ResNet-style)

Scripts: Final_CNN_Images_Custom.ipynb and Final_CNN_Images_Resnet18.ipynb trained on the OpenLogo dataset.

- **Objective:** classify a logo image into a brand class (e.g., Google, Chase, PayPal).
- **Output weights:** best_custom_cnn_image_custom.pth.
- **Label mapping:** class_to_idx_image_custom.json mapping brand name → numeric class id.

This specialist provides an embedding that represents visual brand identity; it enables the fusion model to detect brand mismatch or suspicious visual patterns when paired with text.

6 Phase 2: Multimodal Data Integration

The fusion model requires aligned text and image signals. Phase 2 builds this alignment by producing the Golden Dataset.

6.1 Multimodal Connector

Script: dual_tower_text_features.ipynb.

Inputs: raw email CSVs - CEAS_08.csv, Nazario.csv, Nigerian_Fraud.csv , SpamAssasin.csv and class_to_idx_image_custom.json.

- **Standardization:** cleans and merges raw CSVs into a unified schema (subject, body, label, etc.).
- **Feature engineering:** computes 20 numerical metadata features per email (URLs, sender domain signals, HTML properties, etc.).
- **Critical link:** scans the email body for brand names present in the logo class map; if found, attaches the corresponding image ID.

Output: unified_multimodal_text.csv containing:

- email text
- metadata feature vector
- image ID to load from OpenLogo.

6.2 Linking considerations

- If multiple brands are mentioned, a deterministic tie-breaker should be used (e.g., first match, highest confidence match, or 'unknown').
- If no brand is detected, an 'unknown/no-logo' class or a blank image embedding can be used.
- Text scanning should be case-insensitive and robust to punctuation and obfuscation (e.g., 'Pay-Pal', 'Pa yPal').

7 Phase 3: Fusion Model Training

Phase 3 trains the final model that fuses the three modalities. The model uses the pre-trained specialists as feature extractors and learns a fusion head that maps combined representations to a phishing probability.

7.1 Inputs and data loading

- **unified_multimodal_text.csv** (text, metadata, image ID).
- **vocab_text_1.json** (token mapping).
- **best_custom_cnn_text_1.pth** (text tower weights).
- **best_custom_cnn_image_custom.pth** (image tower weights).
- **openlogo/JPEGImages** (logo images referenced by image ID).

7.2 Fusion logic

For each training row, the pipeline tokenizes the email text, loads the referenced logo image, computes metadata features, and produces three embeddings (text, image, metadata). These embeddings are concatenated into a single vector and passed through a classifier (the fusion head).

Final output: best_fusion_model.pth (the production-ready multimodal phishing detector).

7.3 Evaluation plan

Core metrics: Accuracy, ROC-AUC, Confusion Matrix, precision/recall/F1 at operational thresholds, PR Curve.

8 Phase 4: Inference Pipeline

Inference runs on real HTML emails and produces a phishing probability. The pipeline mirrors training-time preprocessing to avoid train-serve skew.

8.1 Preprocessing and prediction

Script: preprocess_html_and_predict.py.

- **Input:** an HTML email file (optionally with embedded images).
- **Extraction:** parse HTML, extract visible text, URLs, sender-domain signals, and embedded/linked images.
- **Logo selection:** select the most relevant logo (e.g., based on detected brand mention or logo classifier confidence).
- **Model call:** run best_fusion_model.pth to output a phishing confidence score (e.g., 99.4% phishing).

8.2 Outputs

- **Primary:** phishing/legitimate confidence score in [0,100].
- **Secondary:** top contributing signals for explainability (text cues, logo match score, metadata anomalies).

9 Deployment

A typical deployment wraps the inference script and model artifacts into a service that receives emails, runs preprocessing and prediction, and returns a score with a decision (allow/quarantine) based on a configured threshold.

9.1 Deployment Strategy

The phishing detection system was deployed as a web application using Hugging Face Spaces with a Streamlit frontend, enabling browser-based access without requiring local installation.

9.1.1 Architecture

The deployment packages the complete inference pipeline into a containerized Hugging Face Space. The Space includes all required artifacts: the trained fusion model (best_fusion_model.pth), vocabulary mapping (vocab_text_1.json), and the inference script (preprocess_html_and_predict.py).
Hugging Face provides serverless infrastructure with automatic scaling and persistent storage for model artifacts.

9.1.2 User Interface

Streamlit provides the interactive frontend with a simple workflow. Users upload HTML email files through a file uploader widget. Upon upload, the system displays a preview of the email content and extracted features. An "Analyze Email" button triggers the inference pipeline, which parses the HTML, extracts text and images, computes metadata features, runs the fusion model, and returns predictions. The interface displays results including the classification label (Phishing/Legitimate), confidence score as a percentage, and a detailed analysis report highlighting detected issues such as suspicious URLs, urgency language, domain mismatches, and excessive capitalization.

9.1.3 Benefits

This deployment strategy provides several advantages: zero infrastructure management overhead, automatic HTTPS and security, shareable public URL for demonstrations and testing, version control through Hugging Face's Git-based system, and easy updates by pushing new model versions. The serverless architecture eliminates concerns about scaling, monitoring, and maintenance while providing a production-quality interface for real-world email classification.

References

[1] Jay Doshi, Kunal Parmar, Raj Sanghavi, Narendra Shekokar. A comprehensive dual-layer architecture for phishing and spam email detection, Computers & Security, Volume 133, 2023, 103378, ISSN 0167-4048.

[2] OpenLogo Dataset. (2018). QMUL-OpenLogo: Open Logo Detection Challenge. Retrieved from https://qmul-openlogo.github.io/

[3] Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. 3rd International Conference on Learning Representations (ICLR).

[4] Zhang, Y., & Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:1510.03820.

[5] N. A. Alam, "Phishing email dataset," Kaggle. [Online]. Available: https://www.kaggle.com/datasets/naserabdullahalam/phishing-email-dataset/

[6] V. Patil, A. Vora, S. Narayan, and A. S. Namburi, "Phishing Email Detection with Multimodal Deep Learning," Project Report, Group 3, Maryland Applied Graduate Engineering, Univ. of Maryland, College Park, MD, USA, 2025.

[7] V. Patil, A. Vora, S. Narayan, and A. S. Namburi, "phish-detection-ml," GitHub, 2025. [Online]. Available: https://github.com/madhu-anila/phish-detection-ml

[8] Deployment Link: https://huggingface.co/spaces/anilawork/phish-detection-ui-final