# **Multimodal Document Classification Report**

### **Exam Title: Supervised Learning Classification Task Candidate: Kailash Pantha Date:April 18 , 2025**

## **Introduction**

## The goal of this supervised learning classification task is to develop a machine learning model capable of categorizing scanned document images into one of five predefined classes. The dataset provided for this task consists of 2500 grayscale document images (in TIF format), with 500 samples per class. These images originate from various paper-based documents and vary in layout and content. Accompanying each image has a corresponding OCR-generated text file that captures the textual content extracted from the image.

## The dataset is structured into separate directories for images and OCR text, each organized by class labels. This setup presents an opportunity to explore different modeling strategies: either to rely solely on visual information from the images, or on textual data from the OCR files, or a combination of both (multimodal learning).

## 

## 

## 

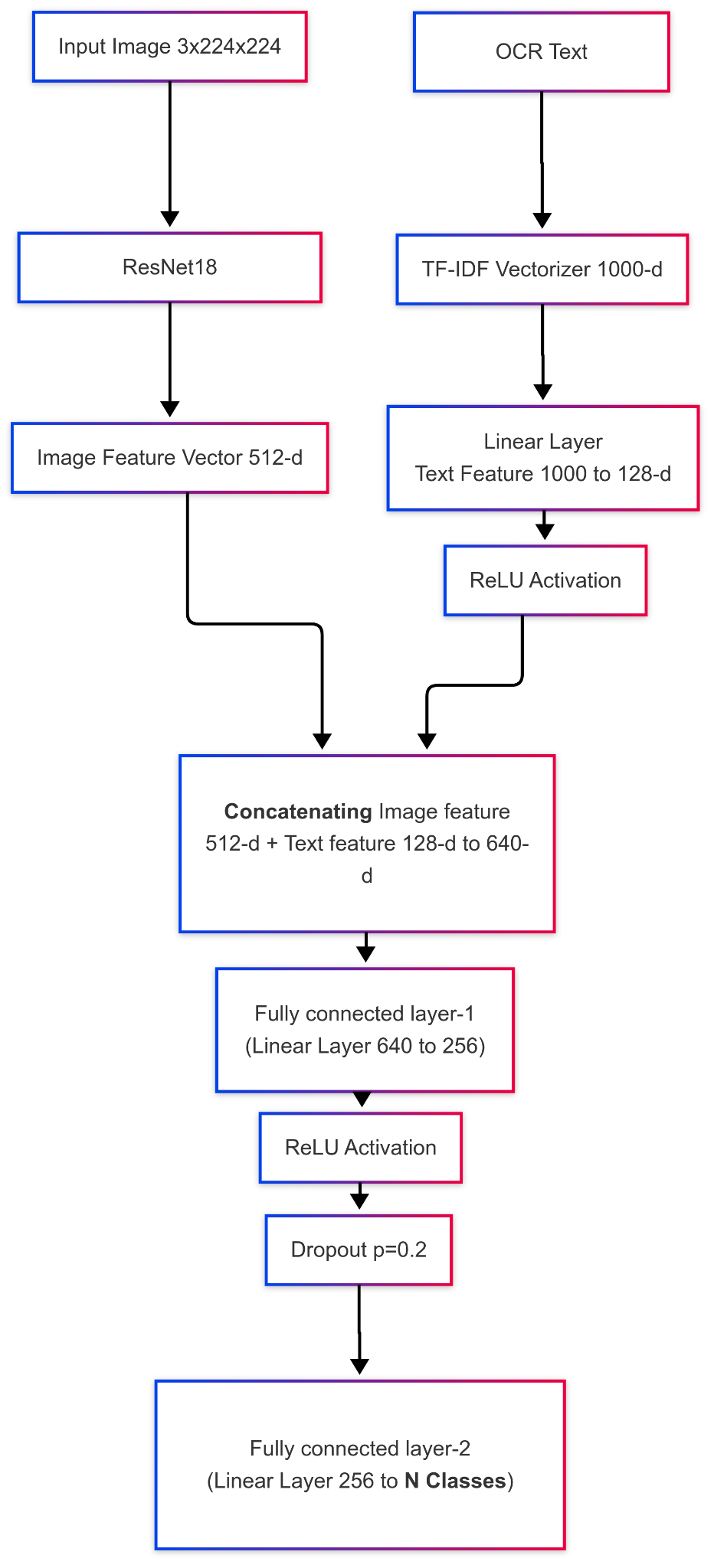
## **Problem Statement**

The objective of this project is to build a supervised machine learning model that can help us to classify scanned document images into one of five predefined categories using either visual features from the images or textual content extracted through OCR or utilizing both features.

## 

## 

## **System Design**



#### 

#### **1. Data Input & Preprocessing**

* **Image Source**: .TIF images categorized in folders by class.
* **OCR Source**: Corresponding .TIF.txt files (extracted text).
* **Classes**: Folder names are used as labels which are given as 0,2,4,6 and 9
* **Preprocessing**:  
  + Images are resized (224×224), normalized, and converted into PyTorch tensors.
  + Text is vectorized using TF-IDF with a fixed vocabulary size (1000).
  + Labels are encoded using LabelEncoder.

#### **2. Dataset Handling**

* A custom Dataset class ImageOCRDataset:  
  + Loads images and their paired OCR text.
  + Applies TF-IDF on OCR text using .set\_tfidf().
  + Supports indexing and batching.
* Dataset is split into **Train (70%)**, **Validation (15%)**, and **Test (15%)** sets using train\_test\_split.

#### **3. Model Architecture: MultiModalClassifier**

* **Image Branch**:  
  + Uses **ResNet18** (pre-trained).
  + The FC layer is removed to get 512-dim image features.
* **Text Branch**:  
  + OCR text vector (1000-dim) → Fully connected layer → 128-dim features.
* **Feature Fusion**: Combining extracted features of both the image files and the corresponding OCR text data and training it through the neural network.  
  + Image (512) + Text (128) features → FC(256) → Dropout → FC(num\_classes).
* **Output**: Class scores via softmax.

#### **4. Training Loop**

* Loss: CrossEntropyLoss
* Optimizer: Adam with learning rate = 0.001
* **Early stopping** with patience = 3 to prevent overfitting.
* Best model (with highest validation accuracy) is saved to best\_model\_v4.pth.

#### **5. Evaluation**

* After training, the model is tested on the test set.
* Computes:  
  + **Accuracy**
  + **Precision / Recall / F1**
  + **Confusion Matrix**
* Results are saved in a report file under ../reports/classification\_statistics.txt.

#### **6. Sample Predictions**

* Visual predictions are shown on a batch from the test loader.

## **Approach Overview**

### **How Did you Solve the Problem?**

To solve the problem of classifying documents using image data or OCR-extracted text, I built a **multimodal deep learning pipeline** that combines visual and textual features.

**A. Understanding the Data**

The dataset included scanned document images and corresponding OCR text files, organized by class. The goal was to leverage both modalities—image layout and text content—for classification into one of five categories.

### **B. Data Preprocessing**

I developed a custom PyTorch Dataset class:

* **Images** were resized (224×224), normalized, and converted to tensors.
* **Text** was transformed using a **TF-IDF vectorizer** (max features = 1000) to capture key semantic features. Each sample returned an image tensor, a TF-IDF vector, and a label.

### **C. Multi- Model Architecture**

I implemented a MultiModalClassifier with two branches:

* An **image branch** using pretrained **ResNet-18** (head removed) to extract 512-d visual features.
* A **text branch** using a linear layer to project TF-IDF vectors into a 128-d space. These were concatenated and passed to a **multi-layer perceptron (MLP)** for final classification into five classes.

### **D. Training Strategy**

I split the data into train/val/test sets using stratification. For training:

* Used **CrossEntropyLoss** and **Adam optimizer**.
* Applied **early stopping** based on validation accuracy.
* Monitored metrics and saved the best-performing model.

### 

### **E. Evaluation**

On the test set, I assessed the model using:

* **Accuracy**, **Precision**, **Recall**, **F1-score**
* **Confusion matrix** for class-wise performance

This multimodal approach significantly improved classification by learning from both layout (image) and content (text) features.

### 

**2. Why did you choose to solve the problem the way you did?**

While analyzing the dataset, I noticed that the documents fell into five distinct categories such as emails, articles, newspapers, posters, and letters. What stood out was that **each class(0,2,4,6,9) had both unique visual structures and distinct textual patterns**.

For example:

* **Letters** often contained keywords like *“Dear,” “Sincerely,”* and had a consistent layout.
* **Emails** included terms like *“To,” “From,” “CC,”* and followed a formal structure.
* **Newspapers** typically had **multi-column layouts**, while **posters** relied more on images and design elements.

Realizing that neither image nor text alone would capture all the relevant signals, I decided to combine both modalities. The **visual layout** helped differentiate structural formats, while the **OCR-extracted text** captured meaningful keywords.

Hence, I opted for a **multimodal classification approach** that leverages both image features and textual content. This fusion was critical for robust performance on such a diverse set of document types.

**3. What challenges did you face?**

1. **Multimodal Feature Fusion** Combining visual and textual modalities was a key challenge. Designing a model that could effectively learn from both ResNet-extracted image features and TF-IDF text embeddings required experimentation. It was important to ensure that both branches contribute meaningfully to the final classification without one dominating the other.
2. **Dimensionality Handling** The mismatch in feature dimensions—1000 for text (TF-IDF) and 512 for image (ResNet)—posed a risk of overfitting, especially with limited training data. I had to carefully design the fusion layers, use regularization techniques like dropout, and fine-tune the architecture to strike a balance between expressiveness and generalization.
3. **Training Instability & Overfitting** Early signs of overfitting highlighted the need for validation monitoring and early stopping. Tuning the learning rate, batch size, and patience parameter took several iterations. Implementing early stopping and saving the best-performing model based on validation accuracy helped stabilize training and improve test performance.

## **4. If you had more time, what would you do differently?**

**Leverage Advanced Language Models** Replace the TF-IDF text representation with transformer-based encoders like DistilBERT or RoBERTa to capture deeper contextual semantics from OCR-extracted text.

**Comprehensive Hyperparameter Tuning** Use tools like GridSearchCV or Optuna to optimize key parameters—learning rate, optimizer choice (e.g., AdamW), dropout rate, batch size, and the number of TF-IDF or embedding dimensions.

**Improved Multimodal Fusion** Explore advanced fusion strategies such as attention-based mechanisms, including Cross-Modal Transformers (e.g., ViLT, LXMERT) or Bilinear Attention Networks (BAN), for more sophisticated interaction between image and text modalities.

**Fine-Tuning the Visual Backbone**

Right now, I only use the early layers of ResNet-18 for feature extraction. If I had more time, I would train the deeper layers of the network as well to better adjust the model to the specific types of images in the dataset. This would help the model understand the visual features more effectively for the task.

**Handling Custom-Input**

I would also take some time to add the feature and create an end point to that handles the custom input data and classify the image data using our saved model.