# 📘 ReadMe: GenreNet – Parallel ViT Model

This document contains instructions for both training/testing the model (backend) and deploying it via a web application (frontend).

---

## 📑 Table of Contents

1. 🧠 Backend: Training & Testing via Jupyter 2
   - 1.1 Prerequisites 2
     - Tools & Access 2
     - Required Files 2
   - 1.2 Environment Setup 2
   - 1.3 Running GenreNet-Parallel ViT.ipynb 3
   - 1.4 Running GenreNet-BERT.ipynb 3
   - 1.5 Running GenreNet-Product Max.ipynb 3
   - 1.6 Output Files 4
   - ⚠️ File Naming Consistency is Crucial 4
2. 🌐 Frontend: Web Deployment via Streamlit 4
   - 2.1 Environment Setup 4
   - 2.2 ▶️ Running the Web App 5
     - Features of the App 5
   - ⚠️ Model File Consistency 5

---

# 1. 🧠 Backend: Training & Testing via Jupyter

## 1.1 Prerequisites

### Tools & Access

- Jupyter Notebook access
- Python ≥ 3.8
- GPU CUDA access (recommended)
- A virtual environment (optional but cleaner)

### Required Files

- `GenreNet-Parallel ViT.ipynb` – image model training & testing notebook
- `GenreNet-BERT.ipynb` – text model training & testing notebook
- `GenreNet-Product Max.ipynb` – multimodal model testing notebook
- `requirements.txt` – list of packages/libraries used
- `BookCover30 Dataset` folder structured as:

```
bookcover30/
├── 224x224/
│   ├── 0001484524.jpg
│   ├── 006062213X.jpg
│   └── ...
├── book30-listing-test.csv
├── book30-listing-train.csv
├── bookcover30-labels-test.text
└── bookcover30-labels-train.text
```

All images from the dataset are contained in the `224x224` folder. The split dataset (for training and testing) can be found by utilizing the respective `.csv` or `.text` files. The `.csv` is mainly used as it contains all information about the dataset. The `.txt` file on the other hand contains information about the labels only.

Due to the size of the dataset, it is **saved as a `zip` file**, please **extract the contents to a folder named `bookcover30`** to be able to use it.

## 1.2 Environment Setup

You can install all required Python packages from the following command line:

```
pip install pandas numpy opencv-python-headless torch torchvision
transformers Pillow tqdm matplotlib seaborn scikit-learn timm
```

Or you can utilize the `requirements.txt` to install the packages/libraries:

```
pip install -r requirements
```

## 1.3 Running `GenreNet-Parallel ViT.ipynb`

1. Open the `GenreNet-Parallel ViT.ipynb` notebook on your IDE of preference.
2. Update save path as needed in **`cell [8]`**: `output_dir = '/folder/model_name'`
3. Execute all cells in order:
   - Data loading from nested folders
   - Color space conversion
   - Data augmentation
   - Model initialization (ViT)
   - Training loop
   - Testing + performance evaluation
   - Best model saving

## 1.4 Running `GenreNet-BERT.ipynb`

4. Open the `GenreNet-BERT.ipynb` notebook on your IDE of preference.
5. Update save path as needed in **`cell [4]`**: `output_dir = '/folder/model_name'`
6. Execute all cells in order:
   - Data loading from nested folders
   - Text tokenization
   - Model initialization (ViT)
   - Training loop
   - Testing + performance evaluation
   - Best model saving

## 1.5 Running `GenreNet-ProductMax.ipynb`

7. Open the `GenreNet-Product Max.ipynb` notebook on your IDE of preference.
8. Update save path as needed in **`cell [20]`**:
   `plt.savefig(f"saves/graphs/final_conf-matrix.png", bbox_inches='tight')`
9. Execute all cells in order:
   - Data loading from nested folders
   - Model initialization (ViT and BERT)
   - Testing + performance evaluation (Individual models)
   - Fusion policy
   - Testing + performance evaluation (Multimodal model)

## 1.6 Output Files

Each notebook will output a graph (in `.png`) of the confusion matrix based on its performance evaluation. The `GenreNet-ParallelViT.ipynb` and `GenreNet-BERT.ipynb` also outputs `.log` and `.pth` (`.safetensors` for `GenreNet-BERT.ipynb`) files for tracking performance and saving the best model from the training and also `.png` of the training loss.

| File Type | File Directory | Description |
|---|---|---|
| `.log` | `saves/logs/` | Training and validation logs |
| `.pth` | `saves/models/` | Best image model |
| `.safetensors` | `saves/models/` | Best text model |
| `.png` | `saves/graphs/` | Graphs: training loss, confusion matrix |

---

## ⚠️ File Naming Consistency is Crucial

Ensure file names in both saving the model during training and in loading the model during evaluation are **consistent**:

- Saving during training: `torch.save(model.state_dict(), 'filename.pth')`
- Loading during testing: `model.load_state_dict(torch.load('filename.pth'))`

Mismatched filenames will cause loading errors or result in evaluating the wrong weights.

---

## 2. 🌐 Frontend: Web Deployment via Streamlit

This section guides you through deploying the trained GenreNet model via a web interface using Streamlit. The program should be able to run through a local machine.

### 2.1 Environment Setup

Make sure the following files are present in the same directory:

| File Name | Purpose |
|---|---|
| `GenreNet.py` | Main Streamlit deployment code |
| `/model/parallel-vit.pth` | Trained image model (from backend) |
| `/model/bert` | Trained text model |
| `requirements.txt` | List of packages/libraries used |

using `requirements.txt`, run the following command to setup environment:

```
pip install -r requirements
```

### 2.2 ▶️ Running the Web App

**Open a terminal** and navigate to the folder containing your deployment files using:

```
cd path/to/your/project/directory
```

💡This makes sure you are in the correct directory where the files are located.

**Launch the app** with:

```
streamlit run GenreNet.py
```

The app will automatically open in your browser, usually at:

```
http://localhost:8501
```

---

### Features of the App

Once launched, the Streamlit app allows users to:

- **Upload an RGB grocery image** for classification and segmentation.
- **View top-1 and top-3 class predictions**, ranked by confidence.
- **Color-code prediction confidence**:
  - Green: High (>60%)
  - Yellow: Moderate (20–59%)
  - Red: Low (<20%)

---

## ⚠️ Model File Consistency

Ensure the model name in the code matches the one in the file:

```
# Load image model
state_dict = torch.load('models/parallel-vit.pth', weights_only=True,
map_location=torch.device('cpu'))

# Load text model
bert_model = BertForSequenceClassification.from_pretrained('models/bert',
num_labels=30)
```

- These lines can be found in GenreNet.py lines `[261]` and `[266]`, respectively.
- Any filename mismatch will result in a `FileNotFoundError`.
- You can rename the model file or update the `load_state_dict()` path accordingly.

If you have a GPU with CUDA, replace line `[261]` with the following:

```
# Load image model
state_dict = torch.load('models/parallel-vit.pth', weights_only=True,
map_location=device)
```