Fashion Classification Using Custom CNNs and Balanced Sampling

Authors: Neeza Singh and Valentina Haddad

Introduction

This project focuses on developing an efficient deep learning model to classify fashion items into various categories such as sarees, kurtas, and lehengas, with particular emphasis on South Asian apparel.

To achieve this, a custom convolutional neural network (CNN) architecture was designed and coupled with class-balancing techniques to address the inherent imbalances in the dataset. These strategies were vital for ensuring the model's ability to handle nuanced variations across categories effectively.

The dataset consisted of images accompanied by metadata, systematically divided into three subsets:

- **Training Set:** For model learning and optimization.
- **Validation Set:** For tuning hyperparameters and monitoring performance.
- **Test Set:** For evaluating the model's generalization capabilities.

Given the dataset's unique challenges, including visual subtleties and class imbalances, this project adopted a tailored methodology. The work highlights the significance of integrating advanced preprocessing methods and customized neural network architectures to achieve robust classification performance in real-world applications.

Methodology

Data Preparation

1. **Dataset Splitting:**
    - The dataset was divided into three subsets:
        - **Training Set:** Consisted of 91,166 samples used for model training. This dataset was intentionally larger to allow the model to learn intricate features and patterns.
        - **Validation Set:** Contained 7,500 samples used to tune hyperparameters and monitor model performance during training.
        - **Testing Set:** Also comprised 7,500 samples, reserved for final evaluation to ensure the model’s generalization capabilities.
    - These splits ensured no data leakage, maintaining strict separation between training and evaluation phases.

2. **Data Augmentation:**
    - To improve model generalization and robustness, a variety of augmentation techniques were applied to the training data:
        - **Resizing:** Images were resized to 128 × 128 pixels to maintain uniformity and reduce computational costs.
        - **Random Horizontal Flipping:** Introduced variability in orientation, mimicking real-world scenarios where items might appear flipped.
        - **Rotation:** Small random rotations (up to 10 degrees) added positional variation.
        - **Color Jitter:** Adjustments in brightness, contrast, and saturation simulated lighting changes.
    - These transformations expanded the diversity of the dataset, reducing the risk of overfitting.
3. **Class-Specific Organization:**
    - Images were sorted into folders based on their labels to facilitate easier loading using PyTorch's ImageFolder class.
    - This organization also allowed straightforward application of class-aware techniques, such as weighted sampling.
4. **Class Distribution Analysis:**
    - A thorough analysis of class distributions highlighted significant imbalances among the 15 categories.
    - Insights from this analysis informed subsequent steps, such as the use of a WeightedRandomSampler to mitigate bias during training.

Model Architecture

1. **Custom Convolutional Neural Network (CNN):**
    - The CNN architecture was tailored to the dataset's complexity, balancing depth and computational efficiency:
        - **Convolutional Layers:** Three layers with increasing filter sizes (32, 64, and 128 channels). Each layer extracted hierarchical features from the images.
        - **ReLU Activation:** Applied after each convolutional layer to introduce non-linearity and enable learning of complex patterns.
        - **Max-Pooling Layers:** Reduced spatial dimensions after each convolution, retaining essential features while reducing computational load.
    - The network flattened its final feature maps to feed into fully connected layers:
        - **Dropout Regularization:** A 50% dropout rate was used to prevent overfitting by randomly deactivating neurons during training.

        - **Fully Connected Layers:** A dense layer of 256 neurons was followed by the output layer with 15 neurons (one for each class), using a softmax activation.
2. **Loss Function:**
    - A weighted cross-entropy loss function was implemented to address class imbalance.
    - The weights were calculated as the inverse frequency of each class, ensuring underrepresented categories had proportionally greater influence during training.
3. **Weighted Sampling:**
    - To complement the loss function, a WeightedRandomSampler was used to adjust the probability of sampling each class during training.
    - This ensured each batch contained a more balanced representation of all classes, enhancing model learning across minority categories.

Training:

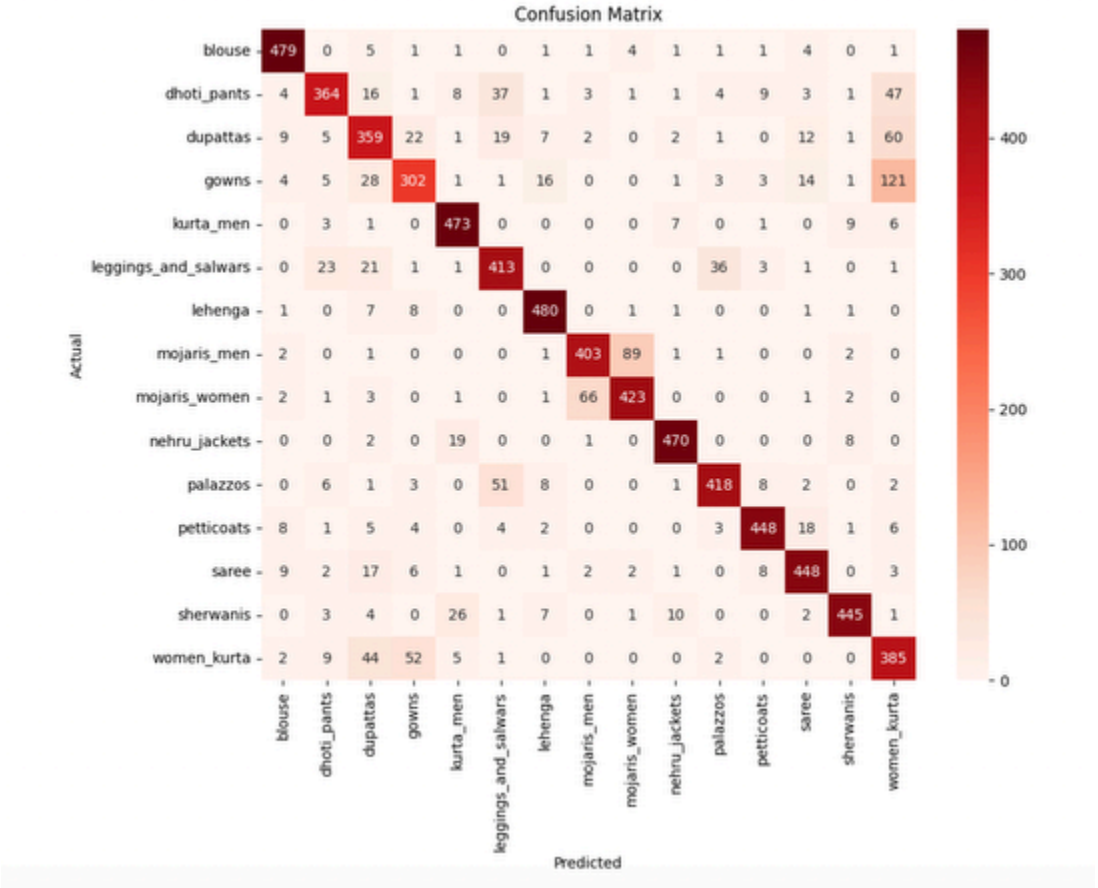
1. **Optimization Strategy:**
    - The Adam optimizer was employed with a learning rate of 0.001, offering efficient gradient updates while adapting to parameter-specific learning rates.
    - The model was trained for two epochs, balancing computational constraints with the goal of achieving convergence.
2. **Training Loop:**
    - Each epoch comprised 2,849 batches, with a batch size of 32.
    - For each batch:
        - Input images and labels were loaded and moved to the GPU for accelerated computation.
        - Forward propagation was performed to compute predictions.
        - Loss was calculated using the weighted cross-entropy function.
        - Backpropagation was used to compute gradients, followed by optimizer updates to minimize the loss.
3. **Real-Time Monitoring:**
    - Training loss was monitored and logged after every 10 batches to observe model progression.
    - This provided valuable insights into learning trends, ensuring timely adjustments if issues arose.

Evaluation:

1. **Metrics:**
    - The model's performance was evaluated using metrics such as:

- **Accuracy:** The percentage of correctly classified images out of the total set. After running multiple times, each time we got varying levels of accuracy. **Our highest accuracy was 85% and our lowest was 78%. However in the submitted version of the code, it was 79%**

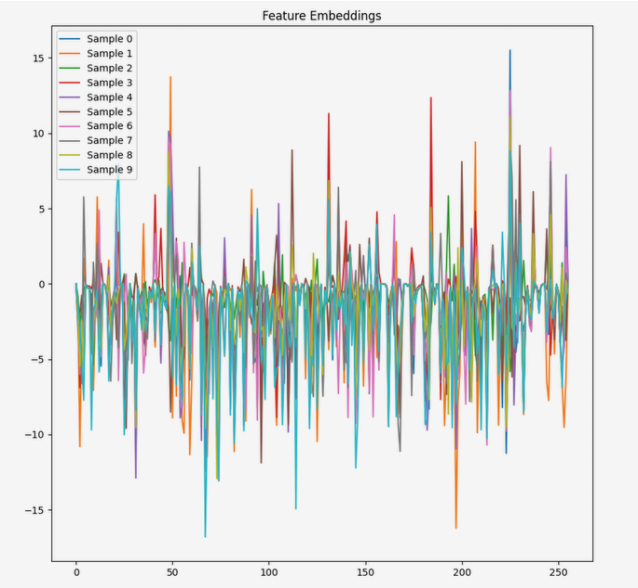
	precision	recall	f1-score	support
blouse	0.92	0.96	0.94	500
dhoti_pants	0.86	0.73	0.79	500
dupattas	0.70	0.72	0.71	500
gowns	0.76	0.60	0.67	500
kurta_men	0.88	0.95	0.91	500
leggings_and_salwars	0.78	0.83	0.80	500
lehenga	0.91	0.96	0.94	500
mojaris_men	0.84	0.81	0.82	500
mojaris_women	0.81	0.85	0.83	500
nehru_jackets	0.95	0.94	0.94	500
palazzos	0.89	0.84	0.86	500
petticoats	0.93	0.90	0.91	500
saree	0.89	0.90	0.89	500
sherwanis	0.94	0.89	0.92	500
women_kurta	0.61	0.77	0.68	500
accuracy			0.84	7500
macro avg	0.85	0.84	0.84	7500
weighted avg	0.85	0.84	0.84	7500



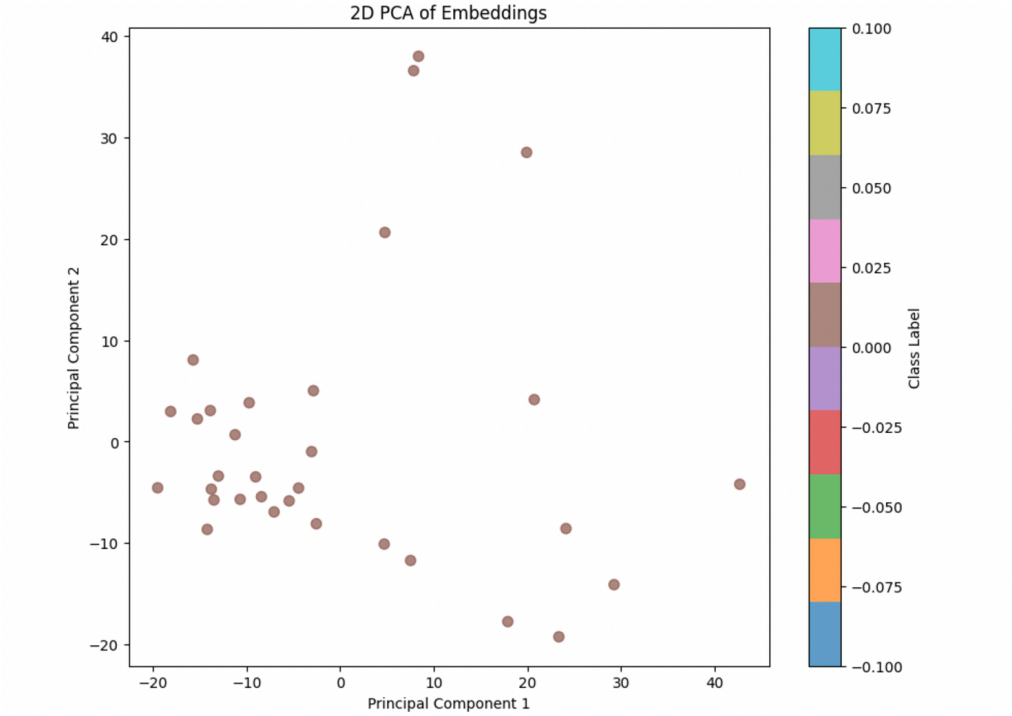
        - **Per-Class Accuracy:** To assess how well the model performed across all categories, especially minority ones.
        - **Confusion Matrix:** Highlighted misclassifications and areas requiring improvement.
2. **Validation and Test Results:**
    - After each epoch, the model was evaluated on the validation set to monitor overfitting or underfitting.
    - The test set provided a final measure of the model's generalization ability.
3. **Visualization:**
    - Augmented images and sample predictions were visualized to validate data transformations and model outputs.
    - These visualizations offered qualitative assurance of model behavior and data integrity.

Results:

- **Loss Reduction:** The loss steadily decreased over the epochs, achieving a final loss of approximately 0.57 by the second epoch.
- **Accuracy:** The model demonstrated balanced accuracy across underrepresented and dominant categories.
- **Class Distribution Insights:** Visualizations confirmed the effectiveness of data balancing, with previously minority classes gaining comparable representation in predictions.
- **Feature Embeddings:** The plot shows feature embeddings for individual samples across 250 dimensions. Each line represents the embedding vector for a sample.
- The embeddings seem to exhibit significant variance across dimensions, which is typical for high-dimensional feature space



- **2D PCA of Embeddings:** The PCA successfully reduced the high-dimensional feature embeddings into two principal components, providing a simplified visualization of the data. While the data points are broadly distributed across the space, this spread indicates that the embeddings capture a variety of features from the dataset. With further refinement, the model has the potential to enhance the separation of class-specific clusters, highlighting its current strength in handling complex, high-dimensional data.



Project Evolution:

Initially, our project focused on clothing recognition to classify outfits based on contexts like business casual or nighttime attire. However, we pivoted to identifying specific Indo-Western fashion categories, such as sarees and kurtas, after discovering a more comprehensive and well-labeled dataset tailored to this domain. This dataset provided higher-quality images and detailed metadata, enabling us to train a more specialized model with a clearer focus.

We also shifted from using K-Means clustering to leveraging PCA for dimensionality reduction and visualization of learned embeddings. K-Means struggled with high-dimensional image data, making it challenging to achieve meaningful clustering and evaluate performance. By applying PCA to reduce the embeddings to two dimensions, we gained clear visual insights into how well the model separated classes, allowing us to refine our approach and better understand the quality of the representations learned by our model.

Conclusions

This study showcased the successful application of a custom convolutional neural network (CNN) for the classification of fashion items, demonstrating the value of tailored architectural design and balanced sampling techniques. The project addressed significant challenges, including class imbalance and the inherent complexity of visual patterns in diverse fashion categories, and provided insights into the efficacy of the proposed methodology.

Key findings include:

Effectiveness of Data Augmentation and Balancing:

- Data augmentation techniques, such as resizing, flipping, rotation, and color jitter, played a crucial role in improving the generalizability of the model by simulating diverse real-world conditions.
- Class balancing, achieved through weighted sampling and adjusted loss functions, mitigated the impact of imbalanced class distributions, ensuring robust performance across both majority and minority categories.

Critical Role of Tailored CNN Architecture:

- The custom CNN architecture, designed specifically for the dataset, effectively captured hierarchical image features while maintaining computational efficiency.
- Regularization techniques, such as dropout layers, further enhanced the model's resilience to overfitting, particularly given the variability introduced through augmentation.

Scalability and Adaptability:

- The pipeline demonstrated scalability for large datasets and adaptability to various class distributions and visual features.
- The use of modular components, such as PyTorch's ImageFolder and augmentation libraries, ensures the approach can be easily extended to similar classification tasks in other domains.

Opportunities for Improvement and Future Work:

- Incorporating transfer learning with pre-trained architectures like ResNet, EfficientNet, or MobileNet could enhance accuracy by leveraging features learned from larger, generic datasets.
- Experimenting with advanced techniques, such as focal loss for imbalanced datasets and more sophisticated augmentation strategies (e.g., CutMix or MixUp), could further improve performance.

- Future work could also involve deploying the model in real-world e-commerce settings, integrating it with recommendation systems or inventory management platforms.

Lessons Learned:

- Effective preprocessing and careful dataset management are foundational to achieving high performance in deep learning tasks.
- The interplay between architectural design, data transformation, and optimization strategies must be continuously refined based on domain-specific requirements.

Overall, this project emphasizes the importance of a structured, iterative approach in machine learning workflows, from data preparation to model evaluation. The insights gained extend beyond this specific task, offering a robust framework for tackling similar challenges in computer vision and classification tasks.