"PatternSense: A Deep Learning Approach to Fabric Pattern Classification"

Vura Hemalatha Kumari
Vejandla Anji
Vemala Sai Kiran
Vemula Madhu

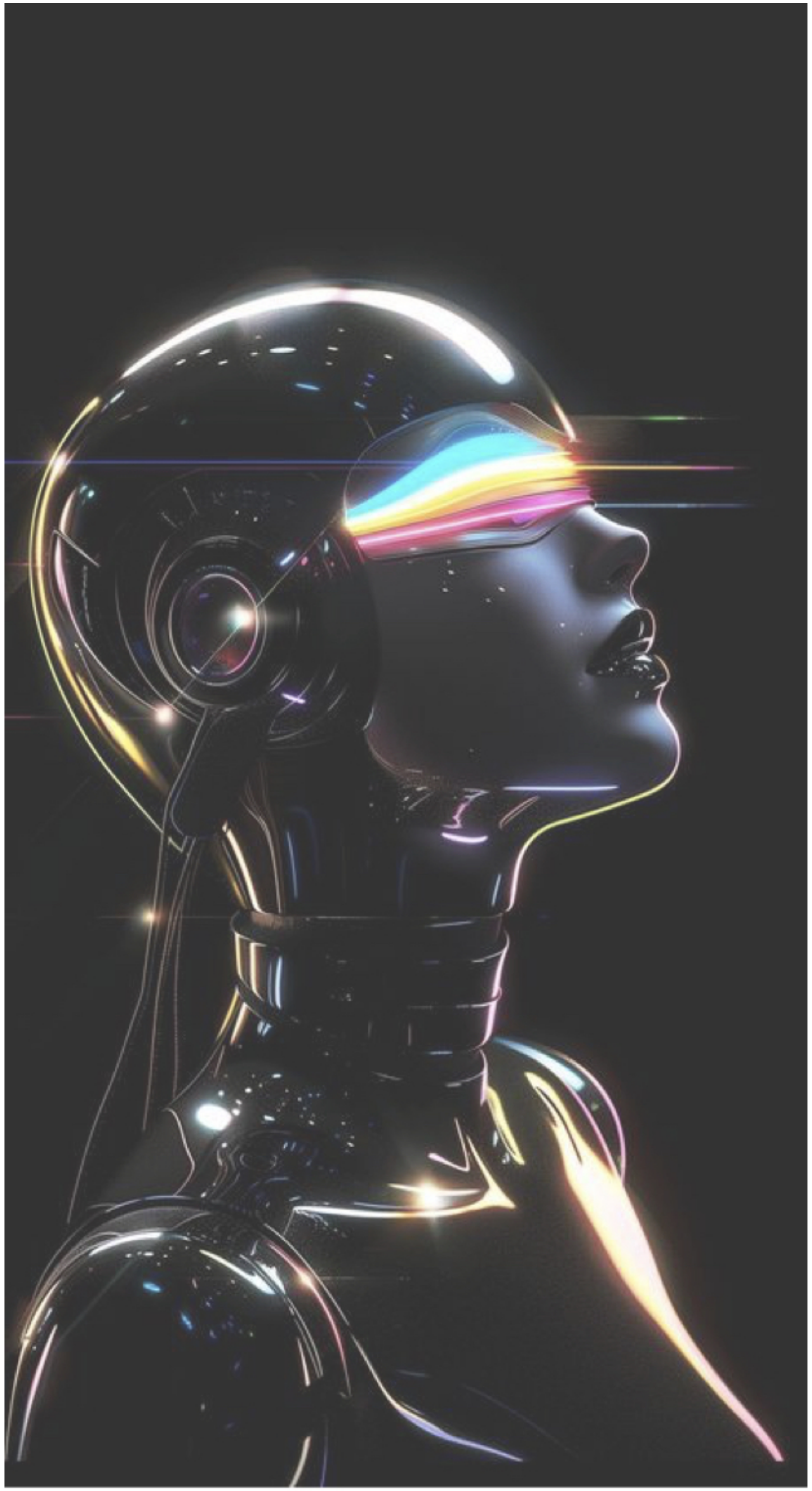
"PatternSense: A Deep Learning Approach to Fabric Pattern Classification"

Introduction:

Deep learning *has transformed computer vision, enabling accurate image classification across various domains. One key application is fabric pattern classification, crucial in fashion, textiles, and e-commerce. Manual pattern recognition is slow and error-prone, highlighting the need for automation.*

PatternSense *is a CNN-based deep learning system developed to classify fabric images into six categories: checkered, dotted, floral, solid, striped, and zigzag. Trained on labeled datasets, it captures unique visual features and ensures high accuracy, even on unseen data.*

Built using ***Python and PyTorch****, the project utilizes techniques like data augmentation and transfer learning to improve performance. PatternSense aims to support industries with smart inventory systems, quality control, and intelligent tagging.*



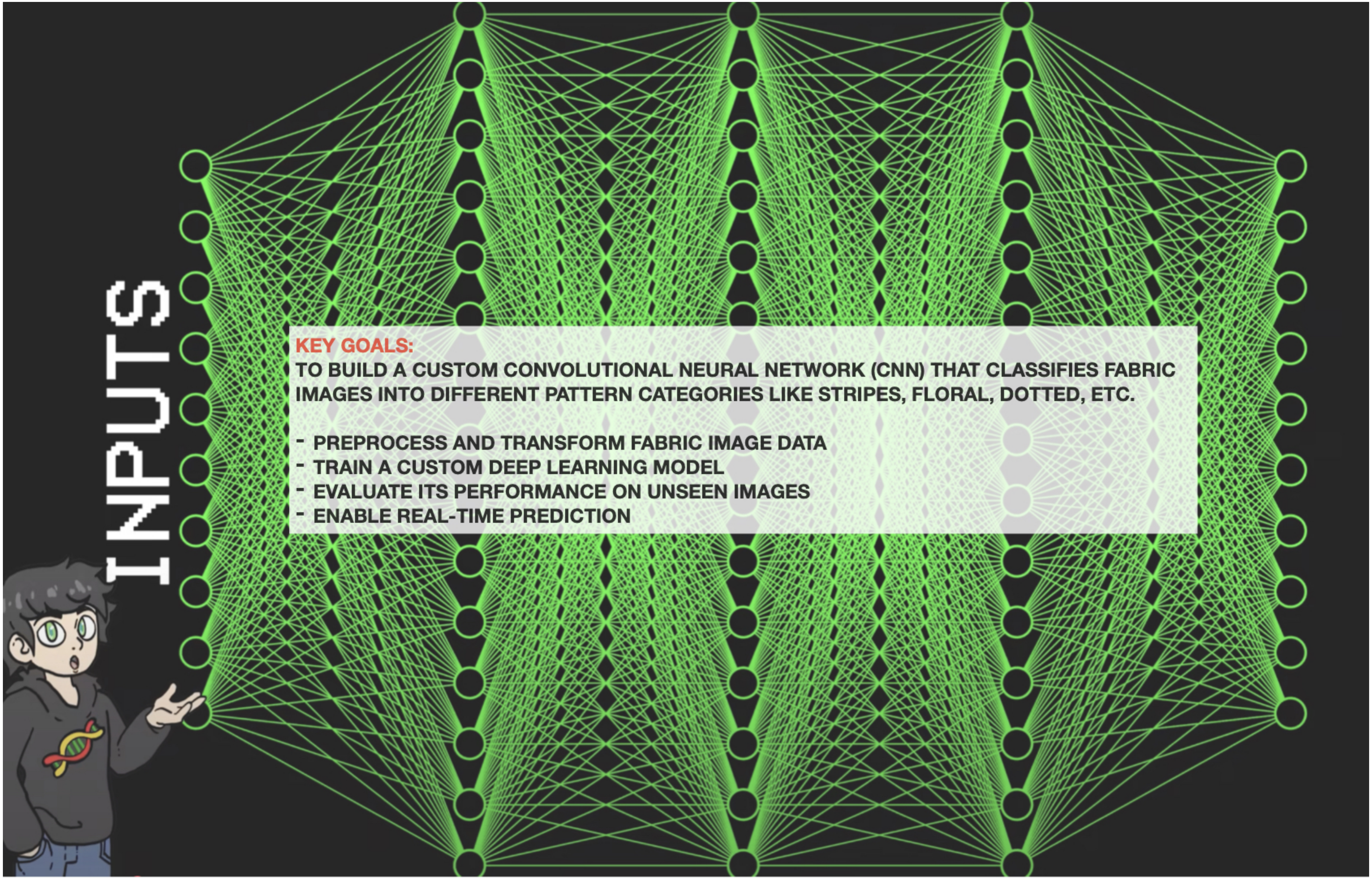
Objectives:

•To develop a CNN-based classifier that accurately identifies fabric patterns from images.

•To create a clean and diverse dataset for different fabric types.

•To evaluate the model's performance using standard metrics like accuracy, precision, recall, and confusion matrix.

•To demonstrate the real-world applicability of the system through testing on new, unseen fabric images.



KEY GOALS:

TO BUILD A CUSTOM CONVOLUTIONAL NEURAL NETWORK (CNN) THAT CLASSIFIES FABRIC IMAGES INTO DIFFERENT PATTERN CATEGORIES LIKE STRIPES, FLORAL, DOTTED, ETC.

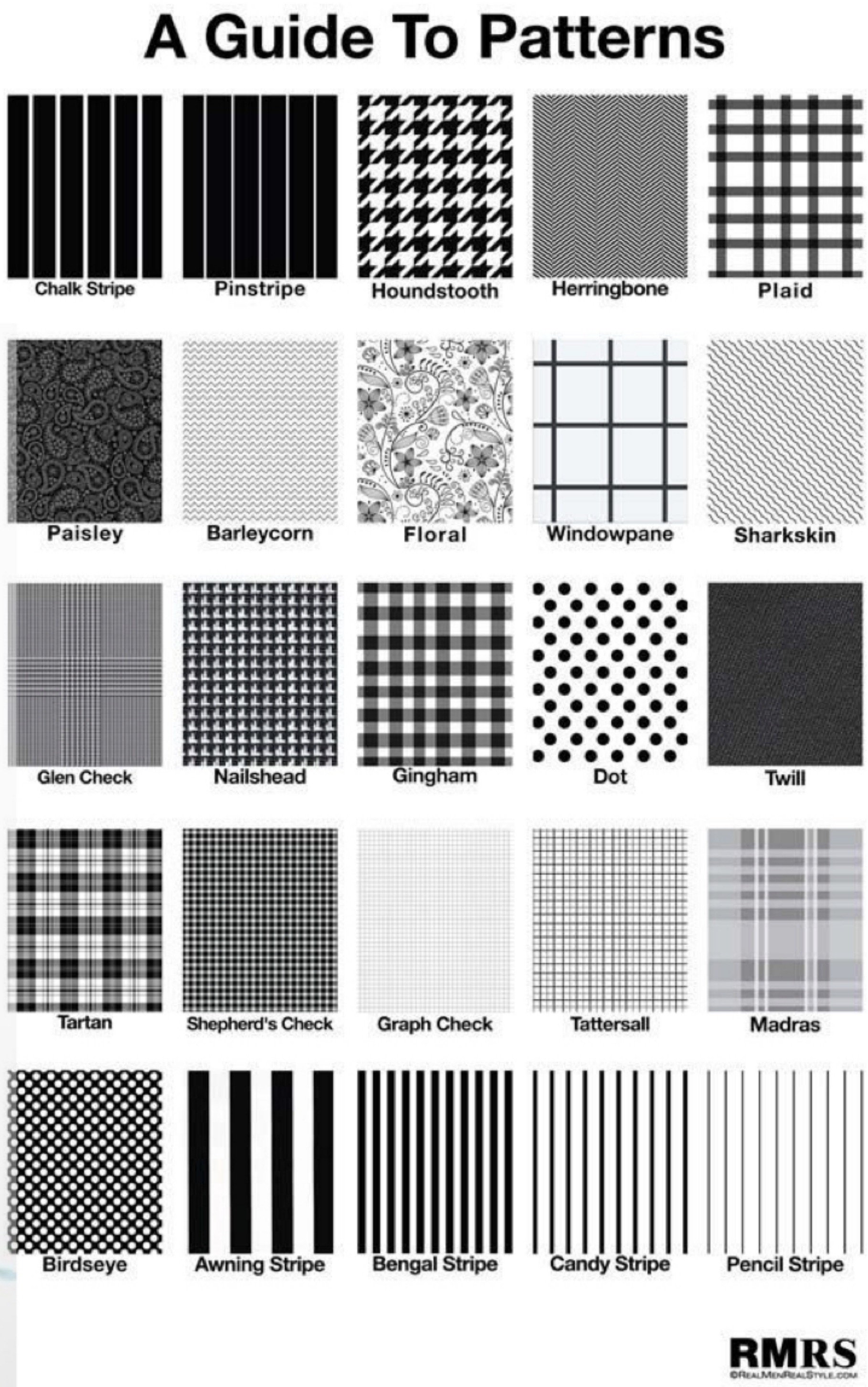
- PREPROCESS AND TRANSFORM FABRIC IMAGE DATA
- TRAIN A CUSTOM DEEP LEARNING MODEL
- EVALUATE ITS PERFORMANCE ON UNSEEN IMAGES
- ENABLE REAL-TIME PREDICTION

Dataset Overview:

Categories:

Checkered
Dotted
Floral
Solid
Striped
Zig Zag
Data Split:

80% Training
20% Validation
(using torch.utils.data.random_split)
Image Size:
128x128 pixels



Model Architecture:

Model Type:

Custom Convolutional Neural Network (CNN)

Architecture:

Conv2D → ReLU → MaxPool

Conv2D → ReLU → MaxPool

Conv2D → ReLU → MaxPool

Flatten → Linear (Dense)

ReLU → Dropout (0.5)

Final Output Layer (num_classes)

Activation Function: ReLU

Loss Function: CrossEntropyLoss

Optimizer: Adam (lr=0.001)

Training Process

Hyperparameters:

Epochs: 10

Batch Size: 32

Image Size: 128x128

Training Loop Includes:

Forward pass

Loss computation

Backpropagation

Optimizer step

epoch to monitor learning progress.

Evaluation & Results
Evaluation Metrics:

```
Confusion Matrix:
[[ 9  3 18  2  5  5  6  9  0 14]
 [ 1  3 12  2  2 10  6  3  5  8]
 [ 3  8 40  1  7  6 15  5  3 17]
 [ 2  3  9  4  7 12 10  5  3 11]
 [ 4  3 16  2  6 10 12  7  1 16]
 [ 1  6  4  8  2 58  7  3  4  3]
 [ 2  1 12  3  1  9 33  8  3  8]
 [ 2  2 14  3 11 14 16 15  2 11]
 [ 0  6  9  4  5 12  6  4 47  7]
 [ 7  5 24  5 12 11  6  8  6 25]]
```

Classification Report:

	precision	recall	f1-score	support
animal	0.29	0.13	0.18	71
cartoon	0.07	0.06	0.07	52
floral	0.25	0.38	0.30	105
geometry	0.12	0.06	0.08	66
ikat	0.10	0.08	0.09	77
plain	0.39	0.60	0.48	96
polka dot	0.28	0.41	0.34	80
squares	0.22	0.17	0.19	90
stripes	0.64	0.47	0.54	100
tribal	0.21	0.23	0.22	109
accuracy			0.28	846
macro avg	0.26	0.26	0.25	846
weighted avg	0.28	0.28	0.27	846