SkyNet: Satellite Image Classification with Deep Learning

Welcome to our project, SkyNet. We'll explore satellite image classification using Convolutional Neural Networks (CNNs), uncovering its potential for environmental monitoring and other applications.

by PREKSHA SHAH 2348446

The Challenge: Understanding Earth from Above

Satellite Images

Complex data capturing Earth's surface.

Classification Goal

Categorize images into predefined classes.

Applications

Environmental monitoring, agriculture, disaster response.

Dataset: Our Window to Earth

1 **20,000 Images**

Diverse categories: water, urban, desert, forest, mountains.

2 **Image Dimensions**

128x128 pixels, RGB format.

3 **Data Preprocessing**

Resizing, normalization, data augmentation.

Preprocessing: Preparing the Data

1. **Resizing**

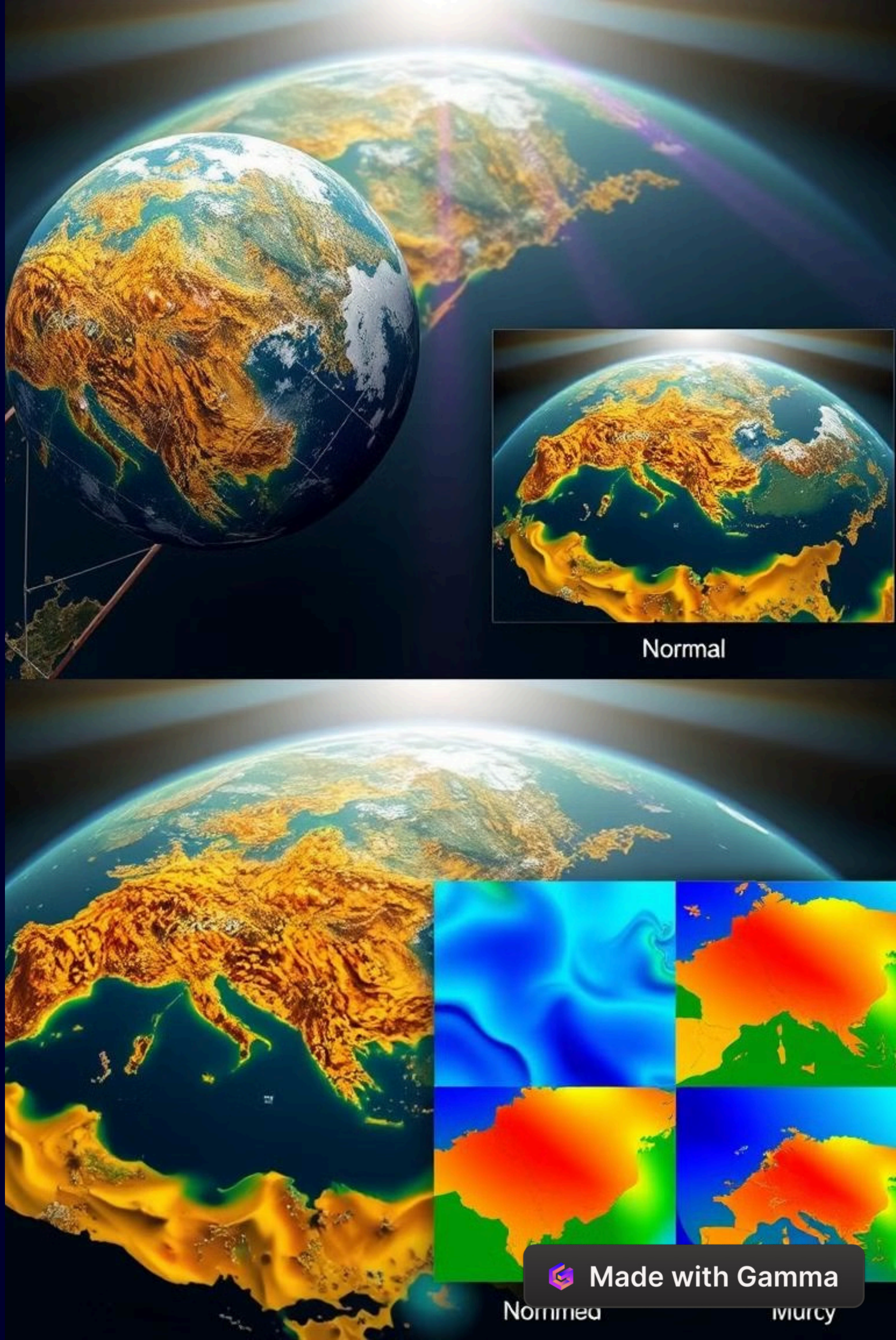
   Ensures uniform image size for consistent model input.

2. **Normalization**

   Scales pixel values for faster model convergence.

3. **Data Augmentation**

   Artificial increase in dataset size to reduce overfitting.



CNN Architecture: The Deep Learning Engine

1

Convolutional Layers

Extract features like edges and textures.

2

Pooling Layers

Downsampling for computational efficiency.

3

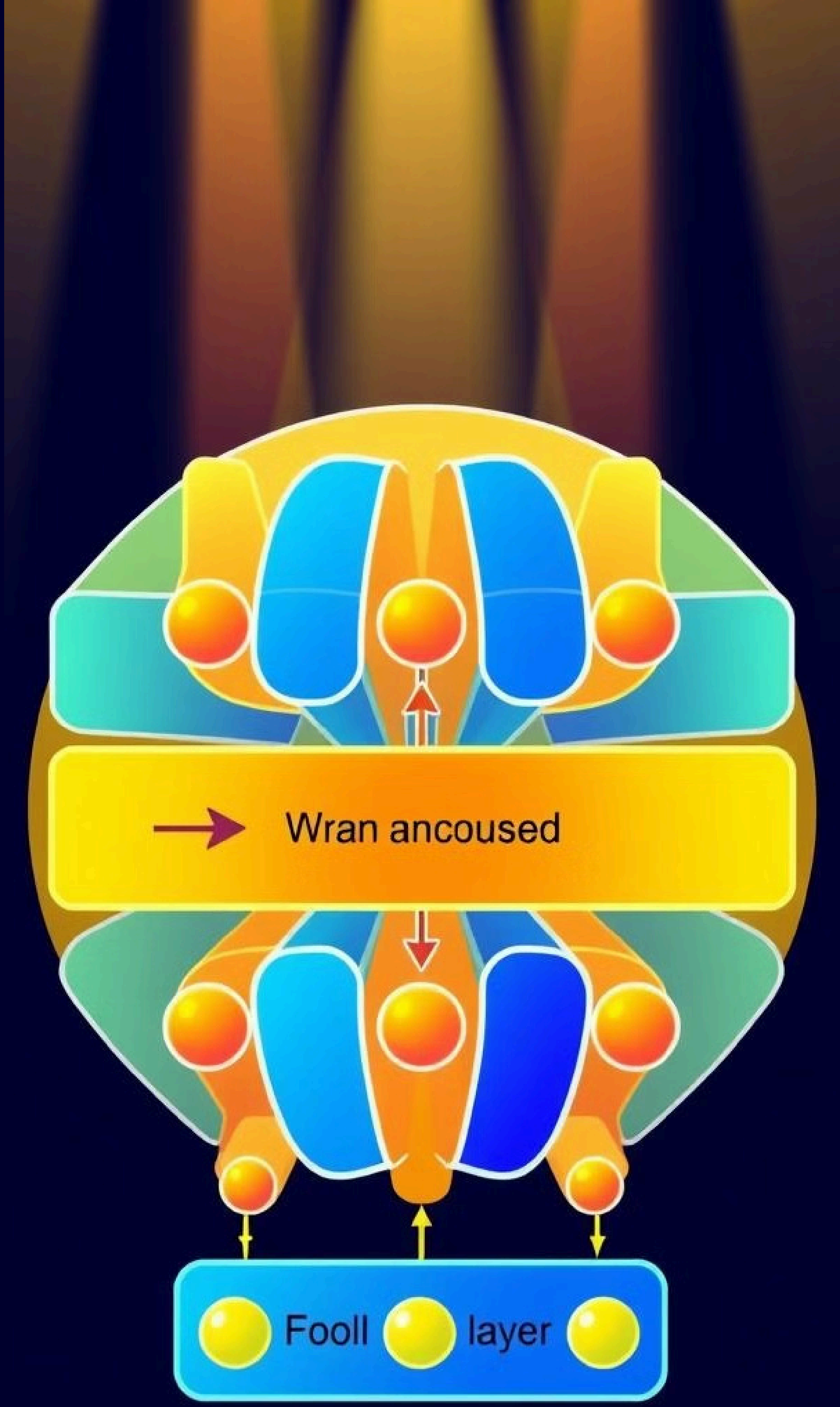
Dropout Layer

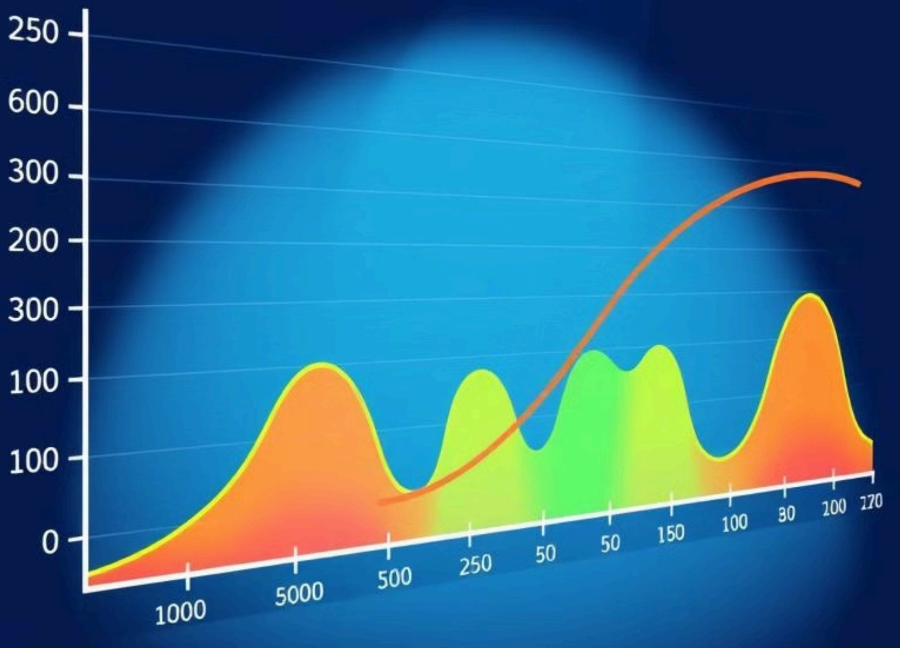
Prevents overfitting by randomly deactivating neurons.

4

Fully Connected Layer

Maps extracted features to final classification.





Training: Teaching the CNN to See

Loss Function

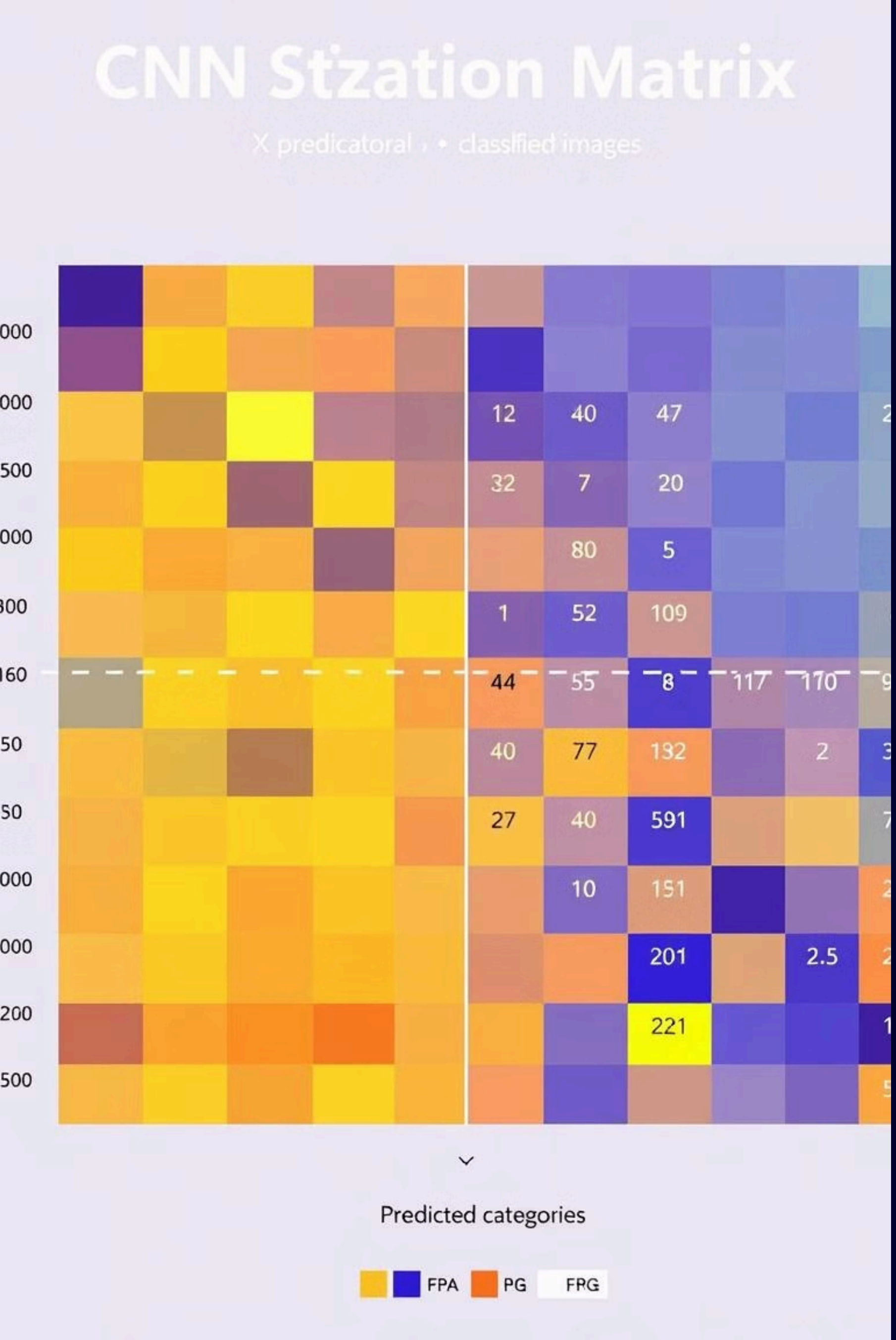
Categorical cross-entropy for multi-class classification.

Optimizer

Adam optimizer with adaptive learning rate.

Metrics

Accuracy and validation loss for model evaluation.



Evaluation: Measuring the CNN's Performance

Overall Accuracy	92%
Water	Highest Accuracy
Urban	Occasionally Misclassified as Desert

Deployment: Making the Model Accessible

Flask Framework

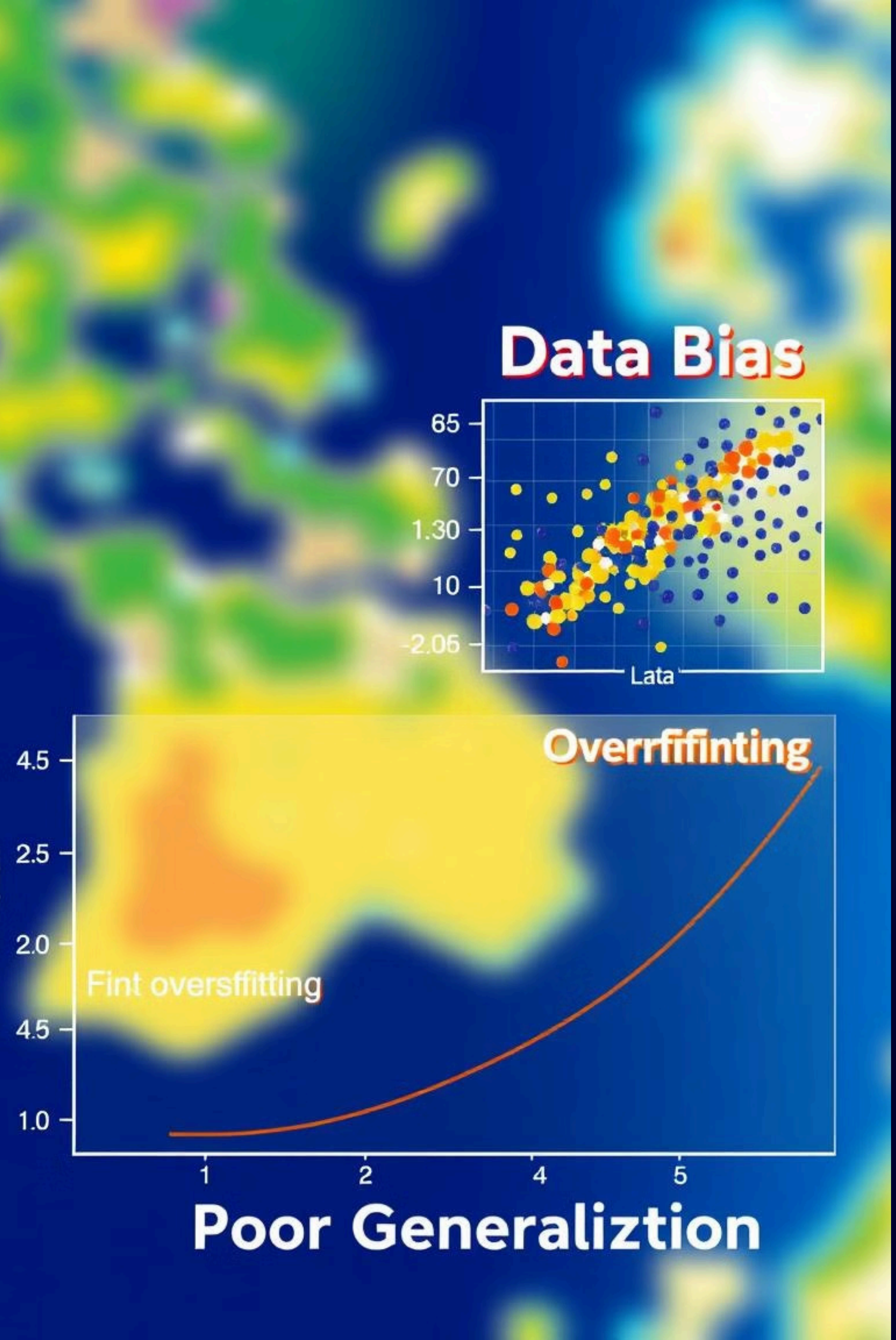
Creates a lightweight API for serving predictions.

Image Upload

API accepts satellite images as input.

Category Prediction

Returns the predicted category for the input image.



Challenges and Solutions

1 **Noisy Images**

Data augmentation techniques were used to mitigate noise.

2 **Overfitting**

Dropout layers and early stopping were employed to prevent overfitting.

3 **Deployment Optimization**

Model size was optimized for faster prediction performance.

Future Directions

1 **Dataset Expansion**

Incorporate more diverse categories and images.

2 **Transfer Learning**

Utilize pre-trained models like ResNet for accuracy improvements.

3 **Real-Time Video Classification**

Optimize the API for classifying video streams in real-time.

Grow detibate:
transfer learing
Expends ritle of stream cassifction
Expen he Aly
Future devecrapiones:
Optimize I the API file for video stream classiction Iasy Inads fur video ofint the collofication.
4. Fast af loading video