AI TechStack 2025
Week 6 Task

October 4, 2025

1 Animal Faces Classification: Understanding Underfitting, Overfitting, Regularization, Dropout, and Hyperparameter Tuning

Objective

Develop a neural network pipeline using Artificial Neural Networks (ANNs) to classify animal faces into three classes: cat, dog, and wild. Explore concepts of underfitting, overfitting, regularization (L1 and L2), dropout, hyperparameter tuning, and the role of validation sets. Evaluate models on accuracy and generalization, with a focus on achieving high test accuracy while considering computational efficiency.

Task Description

You are provided with a semi-prepared Jupyter notebook (`week6.ipynb`) that includes code for downloading and loading the Animal Faces dataset (`andrewmvd/animal-faces`) from Kaggle. The dataset has `train` and `val` folders; use `val` as the test set. If a validation set is needed, split it from the `train` set.

Implement models step-by-step in the provided cells:

- Build and train an underfitting model (simple architecture, high bias).
- Build and train an overfitting model (complex architecture, high variance).
- Add L1 regularization to a model and observe its effects.
- Add L2 regularization to a model and compare with L1.
- Add dropout layers to prevent overfitting.
- Perform hyperparameter tuning (e.g., learning rate, batch size, epochs).
- Combine techniques for a balanced model with high accuracy.

Use `tensorflow.keras` for building ANNs. Validation sets are optional but recommended for tuning and monitoring overfitting. Plot training histories (accuracy/loss curves) and evaluate on the test set using the provided functions.

To improve efficiency, consider reducing image features (e.g., using HOG from previous weeks) before feeding into the ANN, as the images are 512x512 pixels. Priority is on test accuracy, followed by execution speed.

Finally, include a brief text summary describing your approach, what you learned about each concept, and how you achieved your results.

(a) Cat (b) Dog (c) Wild

Figure 1: Sample images from the dataset: one from each class (cat, dog, wild).

Steps to Follow

1. **Data Preparation:**
   - Run the provided cells to download and load the dataset using `ImageDataGenerator`.
   - Note: `val` folder is the test set; create a validation set from `train` if needed.
   - Optionally, visualize a batch of images to confirm loading.
2. **Underfitting Model (Cell 1):**
   - Design a simple ANN (e.g., few layers, small neurons) that underfits.
   - Train on the train set (no validation needed initially).
   - Observe high bias: poor performance on both train and test.
   - Plot history and evaluate on test set.
3. **Overfitting Model (Cell 2):**
   - Design a complex ANN (e.g., many layers, large neurons) that overfits.
   - Train for many epochs without regularization.
   - Observe low bias but high variance: good on train, poor on test.
   - Plot history and evaluate.
4. **L1 Regularization (Cell 3):**
   - Modify a model (e.g., the overfitting one) with L1 regularizers.
   - Optionally use validation split.
   - Train, plot, and evaluate; observe reduced overfitting.
5. **L2 Regularization (Cell 4):**
   - Similar to L1, but use L2.
   - Compare effects with L1.
   - Train, plot, and evaluate.
6. **Dropout (Cell 5):**

- Add dropout layers to a model.
- Train, plot, and evaluate; note prevention of overfitting.

7. **Hyperparameter Tuning (Cell 6):**
   - Experiment with hyperparameters (e.g., learning rate via `Adam(learning_rate=0.001)`, batch size, epochs).
   - Use callbacks like `EarlyStopping` if needed.
   - Must use validation split here for tuning.
   - Train, plot, and evaluate the best configuration.
8. **Combined Approach (Cell 7 - infty.):**
   - Combine techniques (e.g., regularization + dropout + tuned hyperparameters).
   - Aim for good generalization: similar performance on train/validation/test.
   - Optionally experiment with additional models for higher accuracy.
   - Train, plot, and evaluate.

Evaluation and Submission

Submissions are evaluated based on test accuracy (primary), execution efficiency (secondary), correct implementation of concepts, and quality of plots/evaluations. Include a brief report summary of your learnings and approach. Submit:

- Completed Jupyter notebook with all cells implemented.
- Plots of training histories and test evaluations for each model.
- Brief report your approach, results, what you learned about underfitting/overfitting/regularization/dropout/tuning/v and any improvements.