DTU Compute
Department of Applied Mathematics and Computer Science

Hot Dog or Not: Image Classification Using Convolutional Neural Networks

Jone Egon Steinhoff, Lukas Rasocha, Mads Prip & Petr Boska Nylander

Technical University of Denmark, Department of Applied Mathematics and Computer Science, Kgs. Lyngby, Denmark

1 Objective

Figure 1: Example of images from the training dataset with their corresponding labels.

- The main objective is to develop a model that can accurately classify images into two categories: hotdog or not hotdog

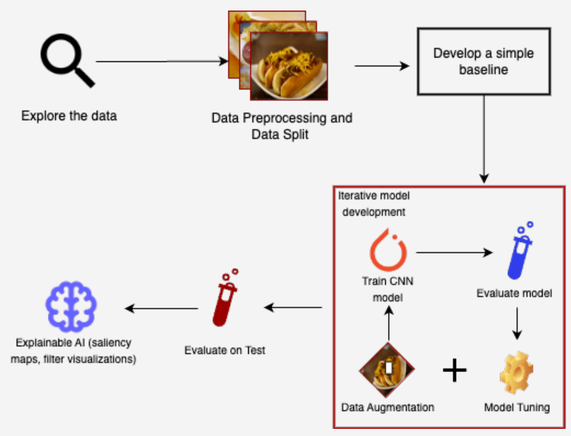

Figure 2: Diagram of the project's pipeline.

2 Data Preprocessing & Augmentation

- Training data split into validation (20%) and training (80 %)
- Data Preprocessing
  - Resize images to 224×224
  - Normalize to $\mu = 0$ and $\sigma = 1$
- Data Augmentation

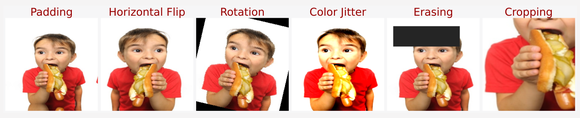

Figure 3: Augmentations applied to the input images. [2]

3 Baseline model

- Simple convolutional neural network (CNN) with 4 convolutional layers

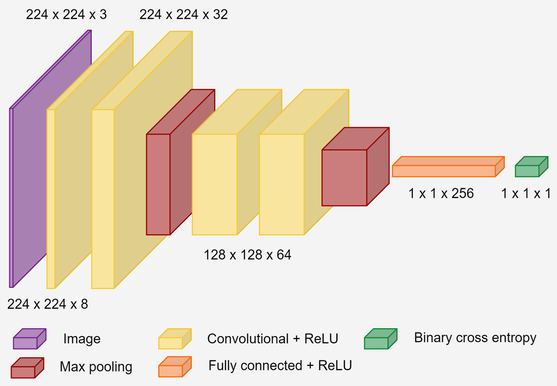

Figure 4: Architecture of the baseline model.

4 Training Process

- Reduce overfitting of the baseline CNN model
- **Convolution setup**: Filter size: 3, Stride: 1, Padding: 1, Max Pooling: 2
- **Training setup:** Batch size: 64, Loss Fun: Binary Cross Entropy, Epochs: Early stopping with patience 5 & Learning Rate: 0.0001 (for other details see diagram below)
- Final model result reported on a holdout test set

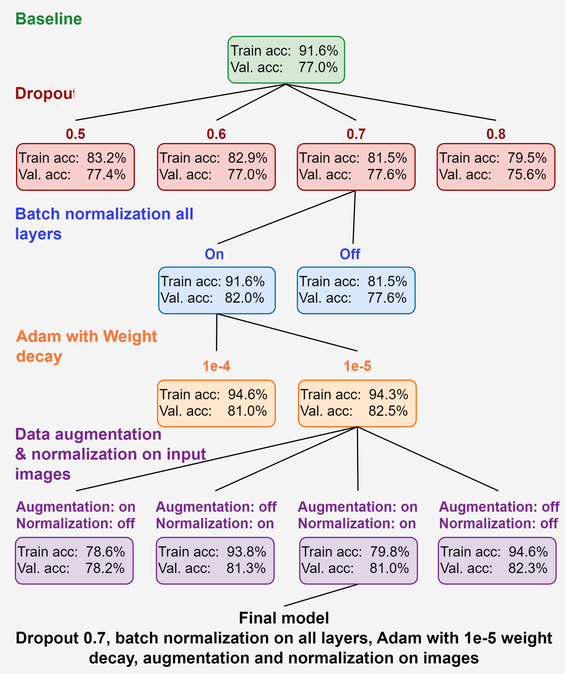

Figure 5: Parameter tuning

5 Challenges & Solutions

- **Overfitting:** Implemented dropout, L2 regularization and data augmentation with early stopping
- **Variance in results:** Results in each model averaged over 5 runs
- **Computational resources:** Limited to not being able to use multiple combinations to find the most optimal set of hyperparameters
- **Slow / bad convergence**: First lowered learning rate of SGD optimizer and later switched to Adam optimizer

6 Results

- Final obtained model:

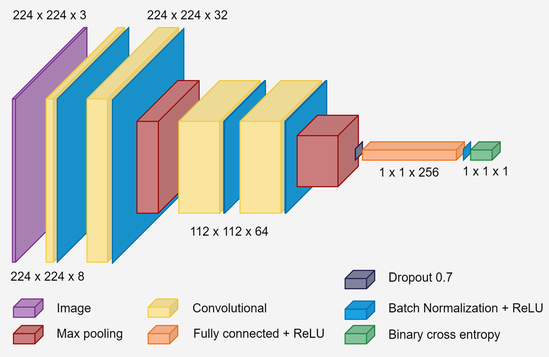

Figure 6: Architecture of the final CNN model.

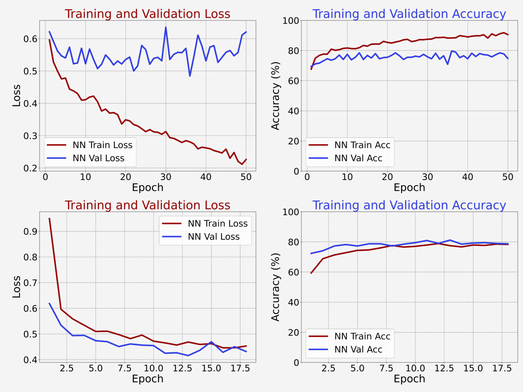

Figure 7: Training and validation loss and accuracies displayed for the baseline (top) and the final model (bottom).

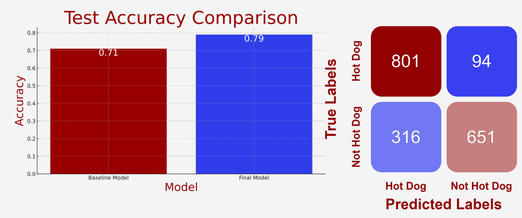

Figure 8: Test accuracy for the baseline model and the final CNN model (left). Confusion matrix on the test set for the final model (right).

7 Conclusion & Learnings

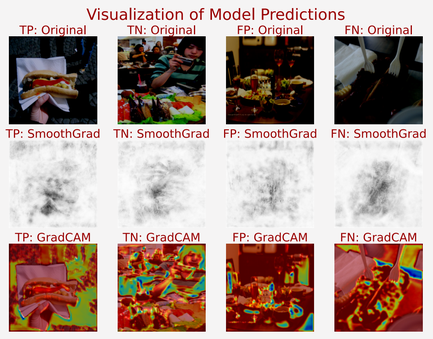

Figure 9: Saliency maps. Top row: normalized original images, middle row: SmoothGrad applied, bottom row: GradCAM applied. [1]

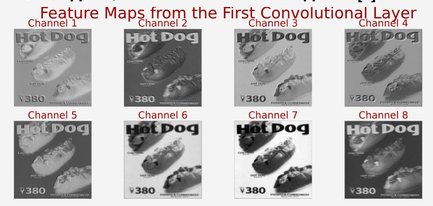

Figure 10: Feature maps from the first CNN layer of a random hot dog image.

- Built a CNN model with a good accuracy for binary image classification
- Learned the importance of data preprocessing and augmentation in improving model performance
- Learned how to debug and optimize deep learning models
- **Future work:** Look into what categories (people, pets, furniture, food) the trained CNN model struggles to predict the most

8 References

- **ChatGPT:** Used for code debugging, more complicated styling of plots, explanation of complicated topics
- 1. Captum: Model Interpretability for PyTorch
- 2. Ultimate Guide to Fine-Tuning in PyTorch : Part 3. Deep Dive to PyTorch Data Transforms with Examples (Ruman, 2023)

DTU Compute
Department of Applied Mathematics and Computer Science
DTU Compute
Department of Applied Mathematics and Computer Science
DTU Compute
Department of Applied Mathematics and Computer Science