

Artificial Vision and Pattern Recognition

Assignment #2: Action Recognition From Still Images Using Deep Learning Networks

Mark Safrónov

MESIIA 2024-2025

The task

Given the dataset in [1], we need to build two image classifiers for the image classes mentioned in this dataset.

One classifier should be a custom-build CNN network, second classifier should be based on a pretrained model like ResNet, GoogLeNet, VGG, MobileNet.

Data preparation

We transform the original dataset into the format readable by the built-in methods from Pytorch, namely, the [torchvision.datasets.ImageFolder](https://pytorch.org/vision/main/generated/torchvision.datasets.ImageFolder.html)¹ class.

The original dataset already includes the suggested train/test split. However, it's a bit unconventional, as there's exactly 100 train images for each image class, and arbitrary amount of test ones, sometimes less than that, sometimes a lot more. So, we prepare two datasets out of the original one.

The first dataset will be called "given splits", and it will be split to train/test data according to the suggestions from the Stanford paper.

The second dataset will contain images split to train and test datasets at random with the classic 80:20 split.

Preliminary analysis of the images in the dataset uncovered the following statistics:

Max Width: 997, Max Height: 965

Min Width: 200, Min Height: 200

Average Width: 424.82364666386906, Average Height: 396.36372219890893

Median Width: 400.0, Median Height: 400.0

Mean: [119.40268332 112.331025 102.14789407], Std: [63.13785612 61.05932064 61.90802032]

We have surprisingly consistent median for the sizes so this will be the basis for resizing the images. We will resize/crop the images to 400×400 pixels in size.

¹ <https://pytorch.org/vision/main/generated/torchvision.datasets.ImageFolder.html>

The mean and std will be used for normalizing the color channels. The data above is the raw value, on the scale of 0..255, we'll need to convert it to the range 0..1

Mean: [0.46824582 0.44051382 0.40057998], Std: [0.24759944 0.23944832 0.24277655]

In addition to using different splits, we'll check whether the bounding boxes we have help in recognizing. We'll use bounding boxes in two ways: as hints for resizing the inputs and as bitmap masks built-in to the images themselves.

CNN model definition

We'll, thus, check the following configurations:

1. given split, random split
2. source images, bounding box-cropped, 4-channel images
3. custom CNN, resnet101

For 4-channel images we'll need to use a fully customized pipeline: a custom data loader, and a customized model.

For cropped images we'll need to only use a custom data loader which crops the images

Construction of the CNN was tool-assisted. The OpenAI's ChatGPT 4o has been used. The following are the explanations given by 4o about the design decisions:

The custom CNN architecture I suggested is inspired by principles and practices that have been established in the field of deep learning through various seminal research papers and widely adopted textbooks. While there isn't a single "definitive source" that outlines the exact architecture I provided, you can refer to several foundational and practical resources to understand why the design choices (e.g., layer arrangement, kernel sizes, and pooling) are effective.

Here are some recommended resources and the relevant insights they offer:

1. Key Papers

AlexNet (Krizhevsky et al., 2012)

- **Paper:** ["ImageNet Classification with Deep Convolutional Neural Networks"](#)
- **Relevance:**
 - Demonstrates the power of deep convolutional networks for image classification.
 - Introduces the use of multiple convolutional and pooling layers followed by fully connected layers.
 - Discusses kernel sizes (e.g., 3x3) and the use of ReLU activations.

- **How It Relates:** The custom CNN mirrors AlexNet's use of stacked convolutional layers, pooling for spatial reduction, and fully connected layers for classification.

VGGNet (Simonyan & Zisserman, 2014)

- **Paper:** ["Very Deep Convolutional Networks for Large-Scale Image Recognition"](#)
- **Relevance:**
 - Advocates for small convolutional kernels (e.g., 3x3) stacked deeper to capture complex features.
 - Uses pooling layers for spatial reduction.
 - Employs fully connected layers for final classification.
- **How It Relates:** The architecture uses 3x3 convolutions and progressively increases the number of filters, following the principles of VGGNet.

ResNet (He et al., 2015)

- **Paper:** ["Deep Residual Learning for Image Recognition"](#)
 - **Relevance:**
 - Introduces residual connections to train very deep networks effectively.
 - Discusses the importance of efficient downsampling (e.g., pooling).
 - Demonstrates the utility of progressively increasing the number of filters.
 - **How It Relates:** While the custom CNN does not include residual connections, it adopts the idea of increasing the depth and number of channels progressively.
-

2. Textbooks

Deep Learning by Ian Goodfellow, Yoshua Bengio, and Aaron Courville

- **Book:** [Deep Learning \(MIT Press, 2016\)](#)
- **Relevance:**
 - Chapters on convolutional networks provide detailed explanations of layer types, pooling, activation functions, and spatial hierarchy.
 - Offers insights into why convolutional architectures work well for image data.
- **How It Relates:** This book explains the reasoning behind convolutional layer stacks, pooling for spatial hierarchy, and fully connected layers.

Dive Into Deep Learning by Aston Zhang et al.

- **Book (Free Online):** [Dive Into Deep Learning](#)
- **Relevance:**
 - Practical, hands-on guide to designing and training CNNs.
 - Discusses common architectures like AlexNet, VGG, and ResNet in detail.
 - Explains why convolutional layers, pooling, and dropout are used in modern architectures.
- **How It Relates:** Covers many of the principles behind the custom CNN, such as 3x3 kernels, pooling layers, and increasing the number of filters.

Neural Networks and Deep Learning by Michael Nielsen

- **Book:** [Neural Networks and Deep Learning](#)
 - **Relevance:**
 - Explains convolutional layers and their role in hierarchical feature learning.
 - Discusses how pooling layers reduce computational complexity and overfitting.
 - **How It Relates:** Provides foundational understanding of the design of convolutional architectures.
-

3. Tutorials and Practical References

Stanford CS231n: Convolutional Neural Networks for Visual Recognition

- **Course Website:** [CS231n: Deep Learning for Computer Vision](#)
- **Relevance:**
 - Lecture notes and assignments provide an in-depth exploration of CNN design.
 - Explains key principles like receptive fields, filter sizes, and pooling.
 - Discusses trade-offs in network depth and complexity.
- **How It Relates:** The concepts of progressively increasing filters and using pooling for downsampling are directly derived from this course.

PyTorch Tutorials

- **Website:** [PyTorch Documentation and Tutorials](#)
 - **Relevance:**
 - Practical examples of designing and implementing custom CNNs.
 - Tutorials often focus on tasks like classification, object detection, and transfer learning.
 - **How It Relates:** Provides practical implementations of architectures similar to the suggested custom CNN.
-

Why This Architecture?

- **Stacked Convolutions (3x3 Kernels):** Borrowed from VGGNet, this allows hierarchical feature extraction.
 - **Pooling for Downsampling:** Reduces spatial dimensions to make the model computationally efficient while retaining essential information.
 - **Increasing Filters:** Inspired by both VGGNet and ResNet, progressively increasing the number of filters allows the model to capture more complex features at deeper layers.
 - **Fully Connected Layers:** Common in traditional architectures for final classification, ensuring global feature aggregation.
 - **Dropout:** Reduces overfitting, as discussed in AlexNet and other research.
-

Summary of Resources

To understand the design of the custom CNN architecture, focus on:

1. **AlexNet and VGGNet papers** for the foundational ideas.
2. **ResNet paper** for insights into progressive depth and filters.
3. **Deep Learning textbooks** for theoretical grounding.
4. **Stanford CS231n** for practical and conceptual understanding.
5. **PyTorch tutorials** for hands-on implementation examples.

By studying these sources, you'll gain a comprehensive understanding of why this architecture works well for your task and how to adapt it further.

Custom CNN has been tailored for **400×400** images.

Pretrained model definition

ResNet 101 has been used as a tradeoff between the computational requirements and the accuracy of predictions

Results analysis

Bibliography

[1] B. Yao, X. Jiang, A. Khosla, A.L. Lin, L.J. Guibas, and L. Fei-Fei. Human Action Recognition by Learning Bases of Action Attributes and Parts. International Conference on Computer Vision (ICCV), Barcelona, Spain. November 6-13, 2011. <http://vision.stanford.edu/Datasets/40actions.html>