Supplementary Material: Designing an Inference Machine for Pose Estimation

Varun Ramakrishna

Introduction

This document details some of the design choices that were made in the design of an inference machine for the task of human pose estimation. We endeavored to be systematic with our design choices and here we explain our experiments that informed our decisions. The inference machine depends on the performance of the classifiers used at each subproblem. Therefore it is imperative to obtain the best possible classifier for each of the subproblems.

Design Choices

In this report we outline our experiments for deciding the following design choices:

- Multi-Class Classifier vs. Multiple Binary Classifiers
- Does Hard Negative mining help?
- HOG features vs. Texture Features
- Random Forests vs. Boosted Random Forests
- Offset Features vs. Patch Features

All experimental results are reported by performing 10-fold cross validation on the training set of the FLIC dataset.

Metrics

Our primary interest is in localizing the parts within the image. Therefore instead of merely classification performance, we use the following localization metrics to measure the performance of our classifiers:

- **Precision/Recall**: We set a threshold of 10 pixels and vary a threshold on the score per part. We perform non-maxima supression to get a set of candidate detections. A detection is scored as a true positive if it lands within the pixel threshold. We plot the precision vs. the recall.
- **Accuracy vs. Pixel threshold**: We measure the number of times the maxima of the scoremap falls within a certain radius of the ground truth. We plot this accuracy as we vary the radius (pixel threshold). We also plot this for more than one detection (top-3 and top-5).

Multi-Class Classifier vs. Multiple Binary Classifiers

The first choice is between using a monolithic multi-class classifier that operates on features extracted from a patch in an image and as output provides a score for each of the classes (in our case, we have $P + 1$ classes, one for each of the P parts in addition to a background class).

At this stage it seems like this decision is still a toss-up. However we choose to use a **multi-part classifier** for it's comparable performance and for it's additional advantages - it is faster to train than P separate classifiers, especially during the *stacking* step.

Does Hard Negative Mining help?

Hard negative mining is a procedure to bootstrap negative samples from a massive negative dataset for localization tasks. The basic idea is to run your initial classifier on your dataset of negative images, find the locations of the most confident misclassifications (hard negatives) and then add them to the training set. This variety of negative sample bootstrapping tends to improve performance of linear classifiers such as SVMs, here we see from figures (1) that all show that performing hard negative mining **improves performance**.

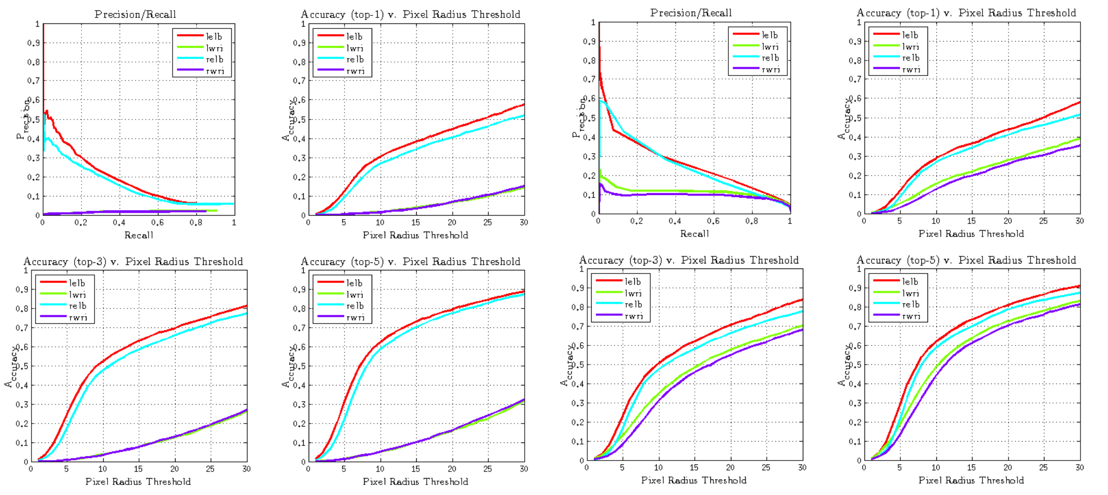

Figure 1: Part localization performance from image features (HOG) on the FLIC dataset using a **multi-class classifier**. The block on the left shows the performance without hard-negative mining. The block on the right shows performance with hard-negative mining.

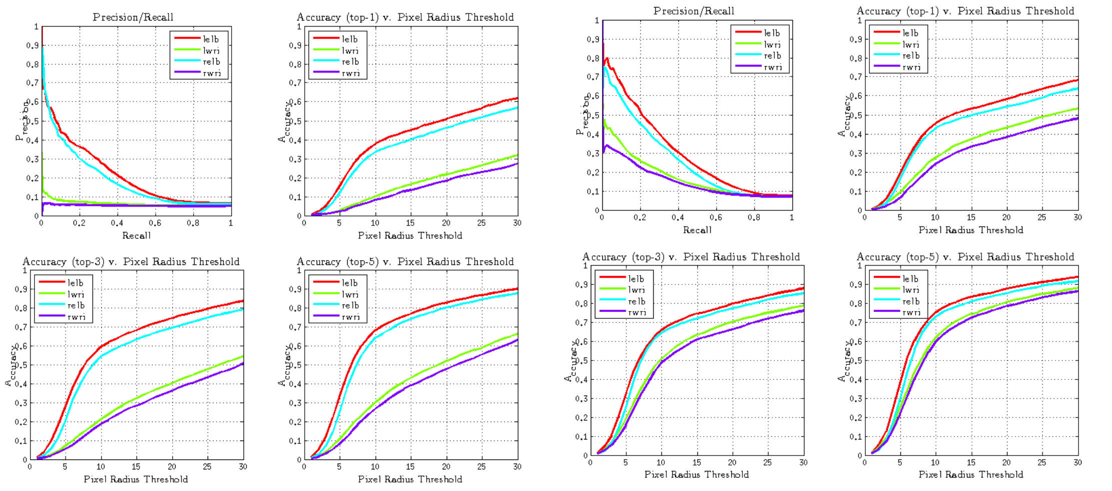

Figure 2: Part localization performance from image features (HOG) on the FLIC dataset using **multiple binary classifiers** - one for each part. The block on the left shows the performance without hard-negative mining. The block on the right shows performance with hard-negative mining.

HOG features vs. Texture features

Figures 3 show that the HOG features perform more or less on par with texture features in the case of the FLIC dataset.

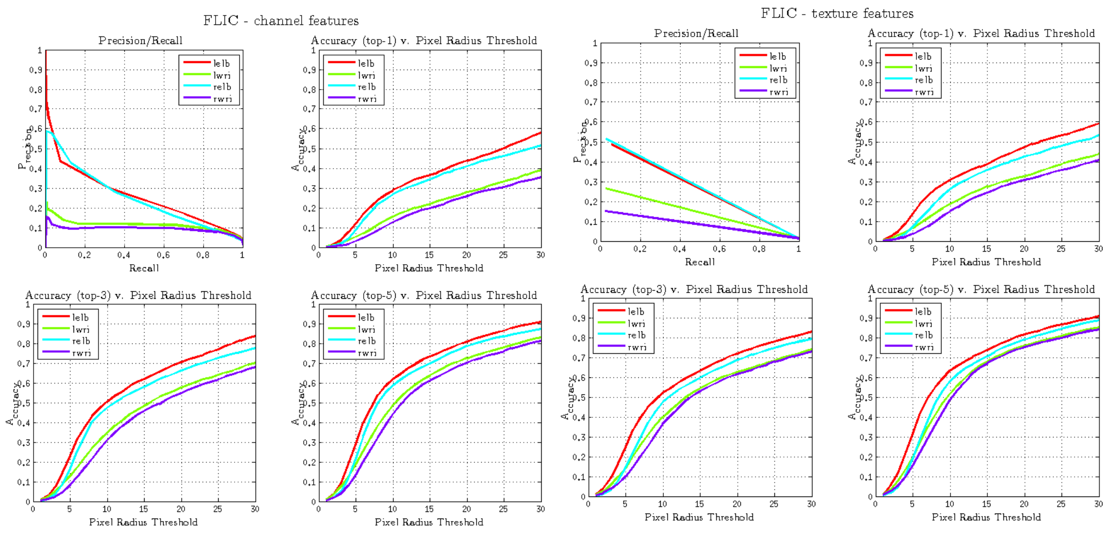

Figure 3: Part localization performance from image features **HOG** on the FLIC dataset using a multi-class classifier. The block on the left shows the performance with **texture features**.

Random Forests vs. Boosted Random Forests

An inference machine is a reduction of structured prediction to a set of sequential prediction problems. The modular nature of the architecture allows us to use any choice of classifier to solve each of these subproblems. We evalute the use of two popular choices of predictor - *random forests* and *boosted random forests*. We use a non-linear classifier as the data is inherently multi-modal and linear classifiers are not particularly suited to handle multi-modality. Figures 4 shows that **boosted random forests outperforms random forests** on the FLIC dataset.

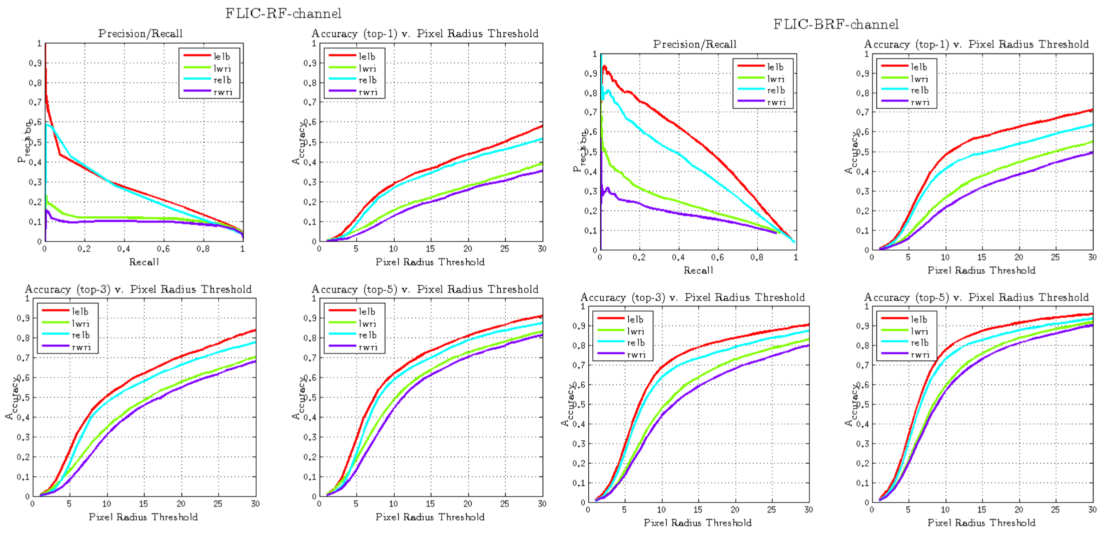

Figure 4: Part localization performance of a multi-part classifier with channel features comparing the use of a random forest (left block) with a boosted random forest (right block).

Offset Features vs. Patch Features

The deeper stages of the inference machines take as input *messages* from the neighboring parts. These messages take the form of features computed on the outputs of each part's scoremap at the location. We evaluate two types of context features - 1) *patch context*, which is essentially concatenated downsampled patches around the location in each scoremap and 2) *peak offsets*, which are offsets to the peaks in each scoremap from the location.

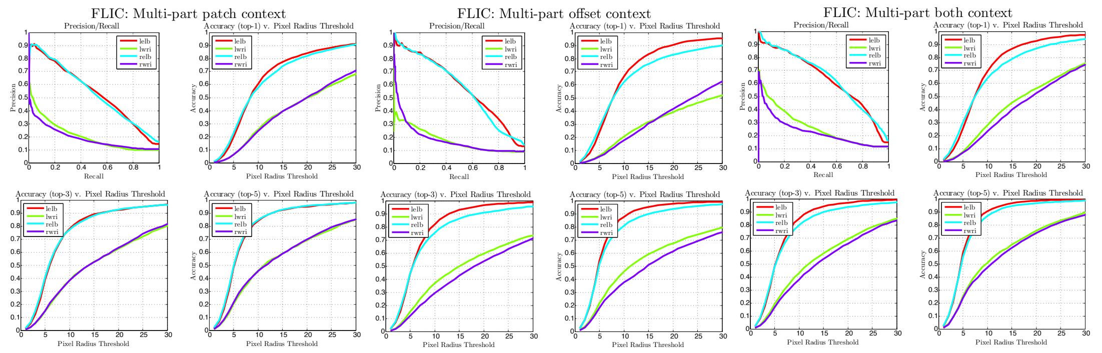

Figure 5: Part localization performance of a 2-stage multipart inference machine with patch features (left) and with peak offset features (right)

We notice that each feature provides an improvement in different regions of the accuracy curves. We also note that **using both features together performs better than using either one**.