Training deformable part models with decorrelated features: 
Supplementary material

Ross Girshick and Jitendra Malik 
UC Berkeley 
{rbg, malik}@eecs.berkeley.edu

1. Introduction

In this technical supplement, we provide two additional studies of DPM training. First, we look at how mean average precision varies as a function of the number of latent positive update iterations. Second, we look at the performance of root-only models (i.e., DPMs without parts) when training with pure latent LDA (LLDA-0), large-margin LLDA (LM-LLDA/LLDA-3), and latent SVM.

2. Latent update iterations

An interesting question when training a DPM is: how many latent update iterations are required to saturate detection performance? To answer this question, we look at mAP plotted against latent update iterations. To make the experimental protocol more precise, we briefly review the DPM training procedure.

DPM training proceeds through three distinct phases. First, single-component, root-only models are trained independently on disjoint subsets of the positive examples (split by aspect ratio and left vs. right facing instances). These root-only models are trained using LSVM or LLDA with the root’s position and scale regarded as latent. Second, the root-only models are combined into a single mixture model, but still without parts. This model is trained using LSVM or LLDA with the choice of component filter, position and scale treated as latent. Finally, parts are added to the model, and the whole model is trained with LSVM or (LM-)LLDA where the component choice, root filter position and scale, and part locations are all latent. Our experiments start from the beginning of the final phase, i.e. the point at which parts are added to the model. To speed up training, we use 200 negative images per classes as was justified in the main paper.

We consider two methods for adding parts. The first method is the default in the voc-release5 source code: parts are added by covering high energy areas of the root filter and then upsampling the root filter weights to twice their original resolution. The second method uses the exact same configuration of parts as the first method, but rather than setting the initial part filter weights by upsampling the root filter, they are all set to zero.

To add intuition to how these initializations change training, consider where parts are placed in the first latent update iteration. When parts are initialized by upsampling, they will move from their resting positions in order to find an optimal tradeoff between their spring deformation cost and how they match the image features. Setting the weights to zero, in contrast, causes the parts to always be placed at their zero-deformation resting positions.

Results are presented in Figure 1. First, focusing on the standard initialization (“upsampled”), we can see that mAP increases by only about 2 points, absolute, from the 1st to the 8th iteration. Both LSVM and LM-LLDA follow similar trajectories. This suggests that one can get away with half the usual number of latent update iterations with no loss in mAP.

To add intuition to how these initializations change training, consider where parts are placed in the first latent update iteration. When parts are initialized by upsampling, they will move from their resting positions in order to find an optimal tradeoff between their spring deformation cost and how they match the image features. Setting the weights to zero, in contrast, causes the parts to always be placed at their zero-deformation resting positions.

Results are presented in Figure 1. First, focusing on the standard initialization (“upsampled”), we can see that mAP increases by only about 2 points, absolute, from the 1st to the 8th iteration. Both LSVM and LM-LLDA follow similar trajectories. This suggests that one can get away with half the usual number of latent update iterations with no loss in mAP.

The second initialization, where part filters are set to zero (“zeroed”) starts from a noticeably lower mAP. In the LSVM case, convergence appears quite slow, and even by the 8th latent update the mAP is still about 1 point lower.
than the other methods. We note that this “zeroed” part initialization was employed by Zhu et al. in [1]. They argue that this initialization is favorable, compared to the standard DPM scheme, because it is simpler. They executed a comparably large number of latent updates—25—in order to saturate mAP. Our results, which show slow convergence, agree with their findings and make the tradeoff between these two initialization choices clear. These results also suggest that an initialization that is even smarter than the somewhat naive upsampling strategy might allow one to stop after only one or two latent updates.

One final remark is that in the “zeroed” case LLDA converges much more quickly than LSVM. We suspect that this is because the LLDA updates are less “sticky” than the LSVM updates in the following sense. When the parts are initialized with zero weights, all positive examples have deformation features that are all zeros. LSVM learns tight springs, because according to the training data positive instances are ones where the parts are never displaced, whereas negative instances are ones where parts might be displaced. The slow convergence of LSVM is likely due to the fact that it needs to overcome this rigidity bias that sticks the parts to where they were first placed.

3. Root-only models

In our second set of experiments we look at training root-only DPMs with LLDA, LM-LLDA, and LSVM. We consider two model choices. The first has one root component and the second has three root components. In both cases, the model has a latent variable specifying if the model is left-facing or right-facing. This latent variable, in effect, doubles the number of mixture components to two and six.

The results are shown in Figure 2. In both cases, LLDA and LSVM have similar mAPs. With either one or three root components, the pure latent LDA method achieves about three-quarters the mAP of the methods using hard negative examples. The three component root-only model trained with LLDA performs about as well as the corresponding model with parts also trained with LLDA (see Figure 2 in the main paper). This observation, coupled with the discriminative calibration results (LLDA-1, Table 1 in the main paper) show that a significant weakness of LLDA training is its inability to calibrate the relative contribution of the different pieces of the model.

References