This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Recombinator Networks: Learning Coarse-to-Fine Feature Aggregation

Sina Honari[1], Jason Yosinski[2], Pascal Vincent[1,4], Christopher Pal[3]
[1]University of Montreal, [2]Cornell University, [3]Ecole Polytechnique of Montreal, [4]CIFAR
[1]{honaris, vincentp}@iro.umontreal.ca, [2]yosinski@cs.cornell.edu, [3]christopher.pal@polymtl.ca

Abstract

Deep neural networks with alternating convolutional, max-pooling and decimation layers are widely used in state of the art architectures for computer vision. Max-pooling purposefully discards precise spatial information in order to create features that are more robust, and typically organized as lower resolution spatial feature maps. On some tasks, such as whole-image classification, max-pooling derived features are well suited; however, for tasks requiring precise localization, such as pixel level prediction and segmentation, max-pooling destroys exactly the information required to perform well. Precise localization may be preserved by shallow convnets without pooling but at the expense of robustness. Can we have our max-pooled multi-layered cake and eat it too? Several papers have proposed summation and concatenation based methods for combining upsampled coarse, abstract features with finer features to produce robust pixel level predictions. Here we introduce another model — dubbed Recombinator Networks — where coarse features inform finer features early in their formation such that finer features can make use of several layers of computation in deciding how to use coarse features. The model is trained once, end-to-end and performs better than summation-based architectures, reducing the error from the previous state of the art on two facial keypoint datasets, AFW and AFLW, by 30% and beating the current state-of-the-art on 300W without using extra data. We improve performance even further by adding a denoising prediction model based on a novel convnet formulation.

1. Introduction

Recent progress in computer vision has been driven by the use of large convolutional neural networks. Such networks benefit from alternating convolution and pooling layers [16, 23, 22, 29, 24, 27, 42] where the pooling layers serve to summarize small regions of the layer below. The operations of convolution, followed by max-pooling, then decimation cause features in subsequent layers of the network to be increasingly translation invariant, more robust, and to more coarsely summarize progressively larger regions of the input image. As a result, features in the fourth or fifth convolutional layer serve as more robust detectors of more global, but spatially imprecise high level patterns like text or human faces [37]. In practice these properties are critical for many visual tasks, and they have been particularly successful at enabling whole image classification [16, 29, 24]. However, for other types of vision tasks these architectural elements are not as well suited. For example on tasks requiring pixel-precise localization or labeling, features arising from max-pooling and decimation operations can only provide approximate localization, as in the process of creating them, the network has already thrown out precise spatial information by design. If we wish to generate features that preserve accurate localization, we may do so using shallow networks without max-pooling, but shallow networks without pooling cannot learn robust, invariant features. What we would like is to have our cake and eat it too: to combine the best of both worlds, merging finely-localized information from shallow, non-pooled networks with robust, coarsely-localized features computed by deep, pooled networks.

Several recently proposed approaches [17, 13, 31] address this by adding or concatenating the features obtained across multiple levels. We use this approach in our baseline model termed *SumNet* for our task of interest: facial keypoint localization. To the best of our knowledge this is the first time this general approach has been applied to the problem of facial keypoint localization and even our baseline is capable of yielding state of the art results. A possible weakness of these approaches however is that all detection paths, from coarsely to finely localized features, only become aggregated at the very end of the feature processing pipeline. As a thought experiment to illustrate this approach's weakness, imagine that we have a photo of a boat floating in the ocean and would like to train a convnet to predict with single pixel accuracy a keypoint corresponding to the tip of the boat's bow. Coarsely localized features[1] could highlight the rough region of the bow of the boat, and finely localized features could be tuned to

[1]From now on we use the shorthand fine/coarse features to mean finely/coarsely localized features.

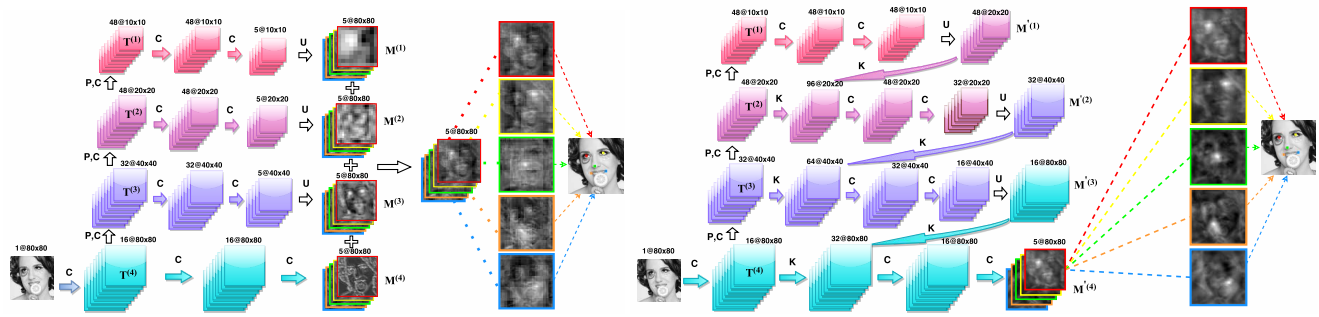

Figure 1. (*Left*) Architecture of summation based coarse-fine network (SumNet). C is a convolutional layer. P,C represents a pooling layer followed by a convolutional layer. All convolutions are 3×3 and all poolings are 2×2. All convolutional layers are followed by ReLU non-linearity except the last convolutional layer in each branch. U represents an upsampling layer. Each branch's output is 5 feature maps of size 80×80. FCN/Hypercolumn models use this architecture. (*Right*) Architecture of the Recombinator Networks (RCN). All convolutions are 3×3 and all poolings are 2×2. All upsamplings are by a factor of 2. K represents concatenation of two sets of feature maps along the feature map dimension. All convolutional layers are followed by ReLU non-linearity except the one right before the softmax. In the Recombinator Networks model with skip connections (not shown), each branch takes upsampled features not only from one coarser branch, but from all coarser branches.

find generic boat edges, but the fine features must remain generic, being forced to learn boat edge detectors for all possible ocean and boat color combinations. This would be difficult, because boat and ocean pixels could take similar colors and textures. Instead, we would like a way for the coarse features which contain information about the global scene structure (perhaps that the water is dark blue and the boat is bright blue) to provide information to the fine feature detectors earlier in their processing pipeline. Without such information, the fine feature detectors would be unable to tell which half of a light blue/dark blue edge was ocean and which was boat. In the *Recombinator Networks* proposed in this paper, the finely localized features are conditioned on higher level more coarsely localized information. It results in a model which is deeper but – interestingly – trains faster than the summation baseline and yields more precise localization predictions. In summary, this work makes the following contributions:

1. We propose a novel architecture — the Recombinator Networks — for combining information over different spatial localization resolutions (Section 3).
2. We show how a simple denoising model may be used to enhance model predictions (Section 4).
3. We provide an in-depth empirical evaluation of a wide variety of relevant architectural variants (Section 5.1).
4. We show state of the art performance on two widely used and competitive evaluations for facial keypoint localization (Section 5.2).

2. Related work

Keypoint localization methods: Our task of interest is the well studied problem of facial keypoint localization [44, 42, 38, 2, 33, 40, 6, 35, 7, 18, 45] illustrated in Figure 1. Precise facial keypoint localization is often an essential preprocessing step for face recognition [1] and detection [45]. Recent face verification models like DeepFace [30] and DeepID2 [27] also include keypoint localization as the first step. There have been many other approaches to general keypoint localization, including active appearance models [8, 43], constrained local models [10, 21, 11, 2], active shape models [12], point distribution models [9], structured model prediction [3, 31], tree structured face models [45], group sparse learning based methods [39], shape regularization models that combines multiple datasets [25], feature voting based landmark localization [26, 36] and convolutional neural networks based models [41, 28, 42]. Two other related models are [31], where a multi-resolution model is proposed with dual coarse/fine paths and tied filters, and [28], which uses a cascaded architecture to refine predictions over several stages. Both of these latter models make hard decisions using coarse information halfway through the model.

Approaches that combine features across multiple levels: Several recent models — including the fully convolutional networks (FCNs) in [17], the Hypercolumn model [13], and the localization model of Tompson et al. [31] — generate features or predictions at multiple resolutions, upsample the coarse features to the fine resolution, and then add or concatenate the features or predictions together. This approach has generally worked well, improving on previous state of the art results in detection, segmentation, and human-body pose estimation [13, 17, 31]. In this paper we create a baseline model similar to these approaches that we refer to as *SumNet* in which we use a network that aggregates information from features across different levels in the hierarchy of a conv-pool-decimate network using concatenation followed by a weighted sum over feature maps prior to final layer softmax predictions. Our goal in this paper is to improve upon this architecture. Differences between the Recombinator Networks and related architectures are sum-

marized in Table 5. U-Net [19] is another model that merges features across multiple levels and has a very similar architecture to Recombinator Networks. The two models have been developed independently and were designed for different problems[2]. Note that none of these models use a learned denoising post-processing as we do (see section 4).

3. Summation versus Recombinator Networks

In this section we describe our baseline SumNet model based on a common architectural design where information from different levels of granularity are merged just prior to predictions being made. We contrast this with the Recombinator Networks architecture.

3.1. Summation based Networks

The SumNet architecture, shown in Figure 1(left), adds to the usual bottom to top convolution and spatial pooling, or "trunk", a horizontal left-to-right "branch" at each resolution level. While spatial pooling progressively reduces the resolution as we move "up" the network along the trunk, the horizontal branches only contains full convolutions and element-wise non-linearities, with no spatial pooling, so that they can preserve the spatial resolution at that level while doing further processing. The output of the finest resolution branch only goes through convolutional layers. The finest resolution layers keep positional information and use it to guide the coarser layers within the patch that they cannot have any preference, while the coarser resolution layers help finer layers to get rid of false positives. The architecture then combines the rightmost low resolution output of all horizontal branches, into a single high resolution prediction, by first up-sampling[3] them all to the model's input image resolution (80×80 for our experiments) and then taking a weighted sum to yield the pre-softmax values. Finally, a softmax function is applied to yield the final location probability map for each keypoint. Formally, given an input image x, define the *trunk* of the network as a sequence of blocks of traditional groups of convolution, pooling and decimation operations. Starting from the layer yielding the coarsest scale feature maps we call the outputs of R such blocks $T^{(1)}, \ldots, T^{(R)}$. At each level r of the trunk we have a horizontal *branch* that takes $T^{(r)}$ as its input and consists of a sequence of convolutional layers with no subsampling. The output of such a branch is a stack of K feature maps, one for each of the K target keypoints, at the same resolution as its input $T^{(r)}$, and we denote this output as $\mathrm{branch}(T^{(r)})$. It is then upsampled $\mathrm{up}_{[\times F]}$ by some factor F which returns the feature map to the original resolution of the input image. Let these upsampled maps be $M_1^{(1)}, \ldots, M_K^{(R)}$ where $M_k^{(r)}$ is the score map given by the r^{th} branch to the k^{th} keypoint (left eye, right eye, ...). Each such map $M_k^{(r)}$ is a matrix of the same resolution as the image fed as input (i.e. 80×80). The score ascribed by branch r for keypoint k being at coordinate i, j is given by $M_{k,i,j}^{(r)}$. The final *probability map* for the location Y_k of keypoint k is given by a softmax over all possible locations. We can therefore write the model as

$$
\begin{aligned}
M^{(1)} &= \mathrm{up}_{[\times 2^{R-1}]}(\mathrm{branch}(T^{(1)})) \\
M^{(2)} &= \mathrm{up}_{[\times 2^{R-2}]}(\mathrm{branch}(T^{(2)})) \\
&\ldots \\
M^{(R)} &= \mathrm{branch}(T^{(R)}) \\
P(Y_k | X = x) &= \mathrm{softmax}\Big(\sum_{r=1}^{R} \alpha_{rk} M_k^{(r)} \Big), \qquad (1)
\end{aligned}
$$

where α_{rk} is a 2D matrix that gives a weight to every pixel location i, j of keypoint k in branch r. The weighted sum of features over all branches taken here is equivalent to concatenating the features of all branches and multiplying them in a set of weights, which results in one feature map per keypoint. This architecture is trained globally using gradient backpropagation to minimize the sum of negated conditional log probabilities of all N training (input-image, keypoint-locations) pairs, for all K keypoints $(x^{(n)}, y_k^{(n)})$, with an additional regularization term for the weights ; i.e. we search for network parameters W that minimize [4]

$$
\mathcal{L}(\mathbf{W}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} -\log P(Y_k = y_k^{(n)} | X = x^{(n)}) + \lambda \|\mathbf{W}\|^2. \qquad (2)
$$

3.2. The Recombinator Networks

In the SumNet model, different branches can only communicate through the updates received from the output layer and the features are merged linearly through summation. In the Recombinator Networks (RCN) architecture, as shown in Figure 1(right), instead of taking a weighted sum of the upsampled feature maps in each branch and then passing them to a softmax, the output of each branch is upsampled, then concatenated with the next level branch with one degree of finer resolution. In contrast to the SumNet model, each branch does not end in K feature maps. The information stays in the form of a keypoint independent feature map. It is only at the end of the R^{th} branch that feature maps are converted into a per-keypoint scoring representation that has the same resolution as the input image, on which a softmax is then applied. As a result of RCN's different architecture, branches pass more information to each other during training, such that convolutional layers in the finer branches get inputs from both coarse and fine layers, letting the network learn how to combine them *non-linearly* to maximize the log likelihood of the keypoints given the

[2] For keypoint localization, we apply the softmax *spatially* i.e. across possible *spatial locations*, whereas for segmentation [13, 17, 19] it is applied across all possible *classes* for each pixel.

[3] Upsampling can be performed either by tiling values or by using bilinear interpolation. We found bilinear interpolation degraded performance in some cases, so we instead used the simpler tiling approach.

[4] We also tried L2 distance cost between true and estimated keypoints (as a regression problem) and got worse results. This may be due to the fact that a softmax probability map can be multimodal , while L2 distance implicitly corresponds to likelihood of a *unimodal* isotropic Gaussian.

input images. The whole network is trained end-to-end by backprop. Following the previous conventions and by defining the concatenation operator on feature maps A, B as $\mathrm{concat}(A, B)$, we can write the model as

$$
\begin{aligned}
&M'^{(1)} = \mathrm{up}_{[\times 2]}(\mathrm{branch}(T^{(1)})) \\
&M'^{(2)} = \mathrm{up}_{[\times 2]}(\mathrm{branch}(\mathrm{concat}(T^{(2)}, M'^{(1)}))) \\
&\dots \\
&M'^{(R)} = \mathrm{branch}(\mathrm{concat}(T^{(R)}, M'^{(R-1)})) \\
&P(Y_k|X = x) = \mathrm{softmax}(M_k'^{(R)}).
\end{aligned} \tag{3}
$$

We also explore RCN with skip connections, where the features of each branch are concatenated with upsampled features of not only one-level coarser branch, but all previous coarser branches and, therefore, the last branch computes $M'^{(R)} = \mathrm{branch}(\mathrm{concat}(T^{(R)}, M'^{(R-1)}, M'^{(R-2)}, \dots, M'^{(1)})$. In practice, the information flow between different branches makes RCN converge faster and also perform better compared to the SumNet model.

4. Denoising keypoint model

Convolutional networks are excellent edge detectors. If there are few samples with occlusion in the training sets, convnets have problem detecting occluded keypoints and instead select nearby edges (see some samples in Figures 3, 5). Moreover, the convnet predictions, especially on datasets with many keypoints, do not always correspond to a plausible keypoint distribution and some keypoints jump off the curve (e.g. on the face contour or eye-brows) irrespective of other keypoints' position (see some samples in Figure 7). This type of error can be addressed by using a structured output predictor on top of the convnet, that takes into account how likely the location of a keypoint is relative to other keypoints. Our approach is to train another convolutional network that captures useful aspects of the prior keypoint distribution (not conditioned on the image). We train it to predict the position of a random subsets of keypoints, given the position of the other keypoints. More specifically, we train the convolutional network as a denoising model, similar to the denoising auto-encoder [34] by completely corrupting the location of a randomly chosen subset of the keypoints and learning to accurately predict their correct location given that of the other keypoints. This network receives as input, not the image, but only keypoint locations represented as one-hot 2D maps (one 2D map per keypoint, with a 1 at the position of the keypoint and zeros elsewhere). It is composed of convolutional layers with large receptive fields (to get to see nearby keypoints), ReLU nonlinearities and no subsampling (see Figure 2). The network outputs probability maps for the location of all keypoints, however, its training criterion uses only prediction errors of the corrupted ones. The cost being optimized similar to Eq.(2) but includes only the corrupted keypoints.

Once, this denoising model is trained, the output of RCN (the predicted most likely location in one-hot binary location 2D map format) is fed to the denoising model. We then simply sum the pre-softmax values of both RCN and denoising models and pass them through a softmax to generate the final output probability maps. The joint model is depicted in Figure 2. The joint model combines the RCN's predicted conditional distribution for keypoint k given the image $P(Y_k|X = x)$ with the denoising model's distribution of the location of that keypoint given other keypoints $P(Y_k|Y_{\neg k})$, to yield an estimation of keypoint k's location given both image and other keypoint locations $P(Y_k|Y_{\neg k}, X = x)$. The choice of convolutional networks for the denoising model allows it to be easily combined with RCN in a unified deep convolutional architecture.

5. Experimental setup and results

We evaluate our model[5] on the following datasets with evaluation protocols defined by previous literature:

AFLW and AFW datasets: Similar to TCDCN [41], we trained our models on the MTFL dataset,[6] which we split into 9,000 images for training and 1,000 for validation. We evaluate our models on the same subsets of AFLW [15] and AFW [45] used by [41], consisting of 2995 and 377 images, respectively, each labeled with 5 facial keypoints.

300W dataset: 300W [20] standardizes multiple datasets into one common dataset with 68 keypoints. The training set is composed of 3148 images (337 AFW, 2000 Helen, and 811 LFPW). The test set is composed of 689 images (135 IBUG, 224 LFPW, and 330 Helen). The IBUG is referred to as the challenging subset, and the union of LFPW and Helen test sets is referred to as the common subset. We shuffle the training set and split it into 90% train-set (2834 images) and 10% valid-set (314 images).

One challenging issue in these datasets is that the test set examples are significantly different and more difficult compared to the training sets. In other words the train and test set images are not from the same distribution. In particular, the AFLW and AFW test sets contain many samples with occlusion and more extreme rotation and expression cases than the training set. The IBUG subset of 300W contains more extreme pose and expressions than other subsets.

Error Metric: The euclidean distance between the true and estimated landmark positions normalized by the dis-

[5] Our models and code are publicly available at https://github.com/SinaHonari/RCN

[6] MTFL consists of 10,000 training images: 4151 images from LFW [14] and 5849 images from the web.

tance between the eyes (interocular distance) is used:

$$\text{error} = \frac{1}{KN}\sum_{n=1}^{N}\sum_{k=1}^{K}\frac{\sqrt{(w_k^{(n)}-\tilde{w}_k^{(n)})^2+(h_k^{(n)}-\tilde{h}_k^{(n)})^2}}{D^{(n)}}, \quad (4)$$

where K is the number of keypoints, N is the total number of images, $D^{(n)}$ is the interocular distance in image n. $(w_k^{(n)}, h_k^{(n)})$ and $(\tilde{w}_k^{(n)}, \tilde{h}_k^{(n)})$ represent the true and estimated coordinates for keypoint k in image n, respectively.

5.1. Evaluation on SumNet and RCN

We evaluate RCN on the 5-keypoint test sets. To avoid overfitting and improve performance, we applied online data augmentation to the 9,000 MTFL train set using random scale, rotation, and translation jittering[7]. We preprocessed images by making them gray-scale and applying local contrast normalization [8]. In Figure S1, we show a visualization of the contribution of each branch of the SumNet to the final predictions: the coarsest layer provides robust but blurry keypoint locations, while the finest layer gives detailed face information but suffers from many false positives. However, the sum of branches in SumNet and the finest branch in RCN make precise predictions.

Since the test sets contain more extreme occlusion and lighting conditions compared to the train set, we applied a preprocessing to the train set to bring it closer to the test set distribution. In addition to the jittering, we found it helpful to occlude images in the training set with randomly placed black rectangles[9] at each training iteration. This trick forced the convnet models to use more global facial components to localize the keypoints and not rely as much on the features around the keypoints, which in turn, made it more robust against occlusion and lighting contrast in the test set. Figure 3 shows the effects of this occlusion when used to train the SumNet and RCN models on randomly drawn samples. The samples show for most of the test set examples the models do a good prediction. Figure 4 shows some hand-picked examples from the test sets, to show extreme expression, occlusion and contrast that are not captured in the random samples of Figure 3. Figure 5 similarly uses some manually selected examples to show the benefits of using occlusion.

To evaluate how much each branch contributes to the overall performance of the model, we trained models excluding some branches and report the results in Table 1. The finest layer on its own does a poor job due to many false positives, while the coarsest layer on its own does a reasonable job, but still lacks high accuracy. One notable result is that using only the coarsest and finest branches together produces reasonable performance. However, the best performance is achieved by using all branches, merging four resolutions of coarse, medium, and fine information. We also experimented with adding extra branches, getting to a coarser resolution of 5×5 in the 5 branch model, 2×2 in the 6 branch model and 1×1 in the 7 branch model. In each branch, the same number of convolutional layers with the same kernel size is applied,[10] and all new layers have 48 channels. The best performing model, as shown in Table 2, is RCN with 6 branches. Comparing RCN and SumNet training, RCN converges faster. Using early stopping and without occlusion pre-processing, RCN requires on average 200 epochs to converge (about 4 hours on a NVidia Tesla K20 GPU), while SumNet needs on average more than 800 epochs (almost 14 hours). RCN's error on both test sets drops below 7% on average after only 15 epochs (about 20 minutes), while SumNet needs on average 110 epochs (almost 2 hours) to get to this error. Using occlusion preprocessing increases these times slightly but results in lower test error. At test time, a feedforward pass on a K20 GPU takes 2.2ms for SumNet and 2.5ms for RCN per image in Theano [4]. Table 2 shows occlusion pre-processing significantly helps boost the accuracy of RCN, while slightly helping SumNet. We believe this is due to global information flow from coarser to finer branches in RCN.

	SumNet		RCN	
Mask	AFLW	AFW	AFLW	AFW
1, 0, 0, 0	10.54	10.63	10.61	10.89
0, 1, 0, 0	11.28	11.43	11.56	11.87
1, 1, 0, 0	9.47	9.65	9.31	9.44
0, 0, 1, 0	16.14	16.35	15.78	15.91
0, 0, 0, 1	45.39	47.97	46.87	48.61
0, 0, 1, 1	13.90	14.14	12.67	13.53
0, 1, 1, 1	7.91	8.22	7.62	7.95
1, 0, 0, 1	6.91	7.51	6.79	7.27
1, 1, 1, 1	**6.44**	**6.78**	**6.37**	**6.43**

Table 1. The performance of SumNet and RCN trained with masks applied to different branches. A mask value of 1 indicates the branch is included in the model and 0 indicates it is omitted (as a percent; lower is better). In SumNet model mask 0 indicates no contribution from that branch to the summation of all branches, while in RCN, if a branch is omitted, the previous coarse branch is upsampled to the following fine branch. The mask numbers are ordered from the coarsest branch to the finest branch.

5.2. Comparison with other models

AFLW and AFW datasets: We first re-implemented the TCDCN model [41], which is the current state of the art

[7] We jittered data separately in each epoch, whose parameters were uniformly sampled in the following ranges (selected based on the validation set performance): Translation and Scaling: [-10%, +10%] of face bounding box size; Rotation: [-40, +40] degrees.

[8] RGB images performed worse in our experiments.

[9] Each image was occluded with one black (zeros) rectangle, whose size was drawn uniformly in the range [20, 50] pixels. It's location was drawn uniformly over the entire image.

[10] A single exception is that when the 5×5 resolution map is reduced to 2×2, we apply 3×3 pooling with stride 2 instead of the usual 2×2 pooling, to keep the resulting map left-right symmetric.

[11] SumNet and RCN models are trained using occlusion preprocessing.

Model	AFLW	AFW
SumNet (4 branch)	6.44	6.78
SumNet (5 branch)	6.42	6.53
SumNet (6 branch)	6.34	6.48
SumNet (5 branch - occlusion)	6.29	6.34
SumNet (6 branch - occlusion)	6.27	6.33
RCN (4 branch)	6.37	6.43
RCN (5 branch)	6.11	6.05
RCN (6 branch)	6.00	5.98
RCN (7 branch)	6.17	6.12
RCN (5 branch - occlusion)	5.65	5.44
RCN (6 branch - occlusion)	**5.60**	**5.36**
RCN (7 branch - occlusion)	5.76	5.55
RCN (6 branch - occlusion - skip)	5.63	5.56

Table 2. SumNet and RCN performance with different number of branches, occlusion preprocessing and skip connections.

Model	AFLW	AFW
TSPM [45]	15.9	14.3
CDM [38]	13.1	11.1
ESR [6]	12.4	10.4
RCPR [5]	11.6	9.3
SDM [35]	8.5	8.8
TCDCN [41]	8.0	8.2
TCDCN baseline (our implementation)	7.60	7.87
SumNet (FCN/HC) baseline (this)	6.27	6.33
RCN (this)	**5.60**	**5.36**

Table 3. Facial landmark mean error normalized by interocular distance on AFW and AFLW sets (as a percent; lower is better).[11]

model on 5 keypoint AFLW [15] and AFW [45] sets, and applied the same pre-processing as our other experiments. Through hyper-parameter search, we even improved upon the AFLW and AFW results reported in [41]. Table 3 compares RCN with other models. Especially, it improves the SumNet baseline, which is equivalent to FCN and Hypercolumn models, and it also converges faster. The SumNet baseline is also provided by this paper and to the best of our knowledge this is the first application of any such coarse-to-fine convolutional architecture to the facial keypoint problem. Figure 6 compares TCDCN with SumNet and RCN models, on some difficult samples reported in [41].

300W dataset [20]: The RCN model that achieved the best result on the validation set, contains 5 branches with 64 channels for all layers (higher capacity is needed to extract features for more keypoints) and 2 extra convolutional layers with 1×1 kernel size in the finest branch right before applying the softmax. Table 4 compares different models on all keypoints (68) and a subset of keypoints (49) reported in [32]. The denoising model is trained by randomly choosing 35 keypoints in each image and jittering them (changing their location uniformly to any place in the 2D map). It improves the RCN's prediction by considering how locations of different keypoints are inter-dependent. Figure 7 compares the output of RCN, the denoising model and the joint model, showing how the keypoint distribution modeling can reduce the error. We only trained RCN on the 2834 images in the train-set. No extra data is taken to pre-train or fine-tune the model [12]. The current state-of-the-art model without any extra data[†] is CFSS[44]. We reduce the error by 15% on the IBUG subset compared to CFSS.

Model	#keypoints	Common	IBUG	Fullset
PO-CR [32]	49	4.00	6.82	4.56
RCN (this)		2.64	5.10	3.88
RCN + denoising keypoint model (this)		**2.59**	**4.81**	**3.76**
CDM [38]	68	10.10	19.54	11.94
DRMF [2]		6.65	19.79	9.22
RCPR [5]		6.18	17.26	8.35
GN-DPM [33]		5.78	-	-
CFAN [40]		5.50	16.78	7.69
ESR [6]		5.28	17.00	7.58
SDM [35]		5.57	15.40	7.50
ERT [7]		-	-	6.40
LBF [18]		4.95	11.98	6.32
CFSS[44]		4.73	9.98	5.76
TCDCN [†] [42]		4.80	8.60	5.54
RCN (this)		4.70	9.00	5.54
RCN + denoising keypoint model (this)		**4.67**	**8.44**	**5.41**

Table 4. Facial landmark mean error normalized by interocular distance on 300W test sets (as a percent; lower is better). [11]

6. Conclusion

In this paper we have introduced the Recombinator Networks architecture for combining coarse maps of pooled features with fine non-pooled features in convolutional neural networks. The model improves upon previous summation-based approaches by feeding coarser branches into finer branches, allowing the finer resolutions to learn upon the features extracted by coarser branches. We find that this new architecture leads to both reduced training time and increased facial keypoint prediction accuracy. We have also proposed a denoising model for keypoints which involves explicit modeling of valid spatial configurations of keypoints. This allows our complete approach to deal with more complex cases such as those with occlusions.

Acknowledgments

We would like to thank the Theano developers, particularly F. Bastien and P. Lamblin, for their help throughout this project. We appreciate Y. Bengio and H. Larochelle feedbacks and also L. Yao, F. Ahmed and M. Pezeshki's helps in this project. We also thank Compute Canada, and Calcul Quebec for providing computational resources. Finally, we would like to thank Fonds de Recherche du Québec – Nature et Technologies (FRQNT) for a doctoral research scholarship (B2) grant during 2014 and 2015 (SH) and the NASA Space Technology Research Fellowship (JY).

[12] We only jittered the train-set images by random scaling, translation and rotation similar to the 5 keypoint dataset.

[†] TCDCN [42] uses 20,000 extra dataset for pre-training.

Features \ Models	Efficient Localization [31]	Deep Cascade [28]	Hyper-columns [13]	FCN [17]	RCN (this)
Coarse features: hard crop or soft combination?	Hard	Hard	Soft	Soft	Soft
Learned coarse features fed into finer branches?	No	No	No	No	Yes

Table 5. Comparison of multi-resolution architectures. The Efficient Localization and Deep Cascade models use coarse features to crop images (or fine layer features), which are then fed into fine models. This process saves computation when dealing with high-resolution images but at the expense of making a greedy decision halfway through the model. Soft models merge local and global features of the entire image and do not require a greedy decision. The Hypercolumn and FCN models propagate all coarse information to the final layer but merge information via addition instead of conditioning fine features on coarse features. The Recombinator Networks (RCN), in contrast, injects coarse features directly into finer branches, allowing the fine computation to be tuned by (conditioned on) the coarse information. The model is trained end-to-end and results in *learned* coarse features which are tuned directly to support the eventual fine predictions.

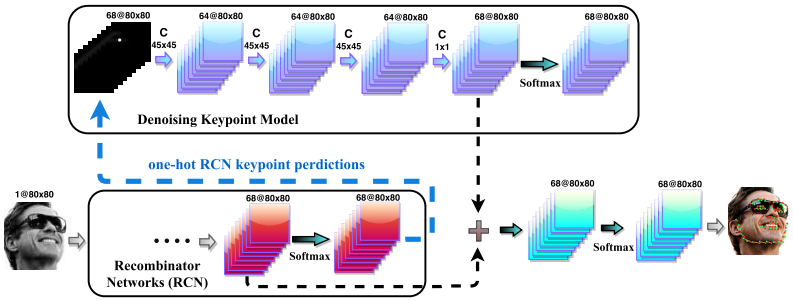

Figure 2. Denoising / joint keypoint model. The Recombinator Networks (RCN) and the keypoint location denoising models are trained separately. At test time, the keypoint hard prediction of RCN is first injected into the denoising model as one-hot maps. Then the pre-softmax values computed by the RCN and the denoising models are summed and pass through a final softmax to predict keypoint locations.

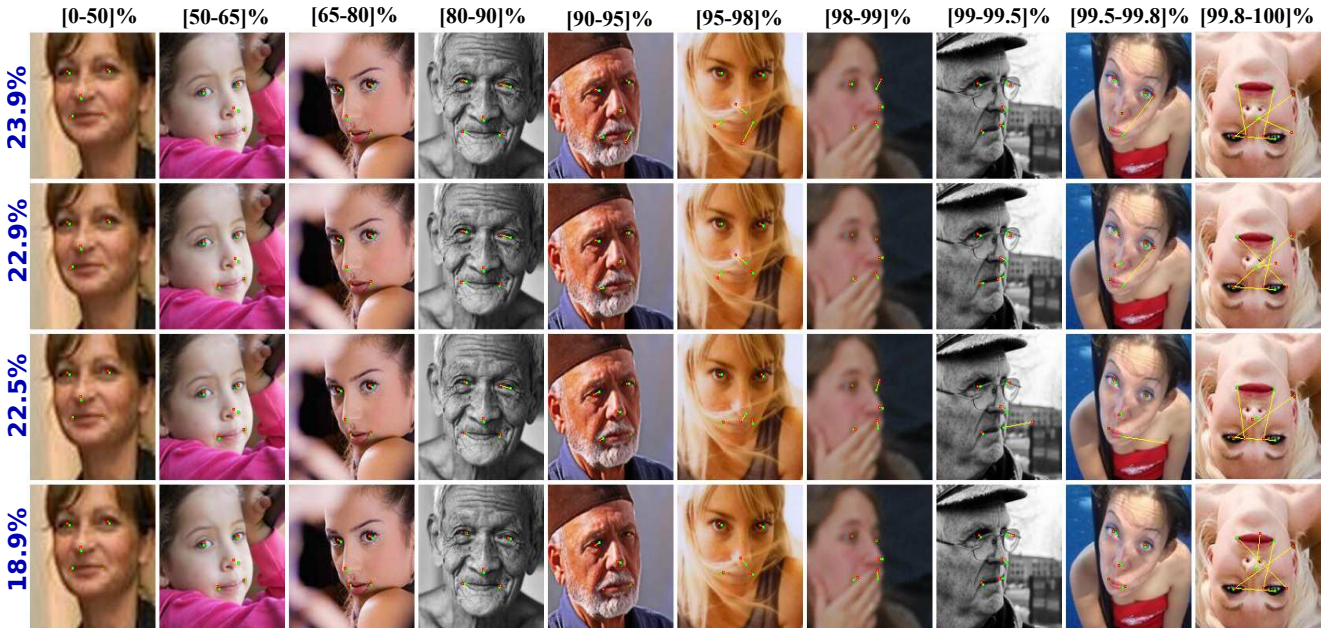

Figure 3. Keypoint predictions on random test set images from easy (left) to hard (right). Each column shows predictions of following models from top to bottom: SumNet, SumNet with occlusion, RCN, RCN with occlusion (all models have 5 branches). We note for each test set image (including both AFLW and AFW) the average error over the four models and use this as a notion of that image's difficulty. We then sort all images by difficulty and draw a random image from percentile bins, using the bin boundaries noted above the images. To showcase the models' differing performance, we show only a few easier images on the left side and focus more on the hardest couple percent of images toward the right side. The value on the left is the average error of these samples per model (much higher than the results reported in Table 3 because of the skew toward difficult images). The yellow line connects the true keypoint location (green) to the model's prediction (red). Dots are small to avoid covering large regions of the image. Best viewed with zoom in color. Figure S2 shows the performance of these four models as the difficulty of the examples increase.

Figure 4. Samples with different expressions (green border), contrast and illuminations (red border) and occlusions (blue border) from AFLW and AFW sets. In each box, top row depicts samples from SumNet and bottom row shows samples from RCN, both with occlusion pre-processing.

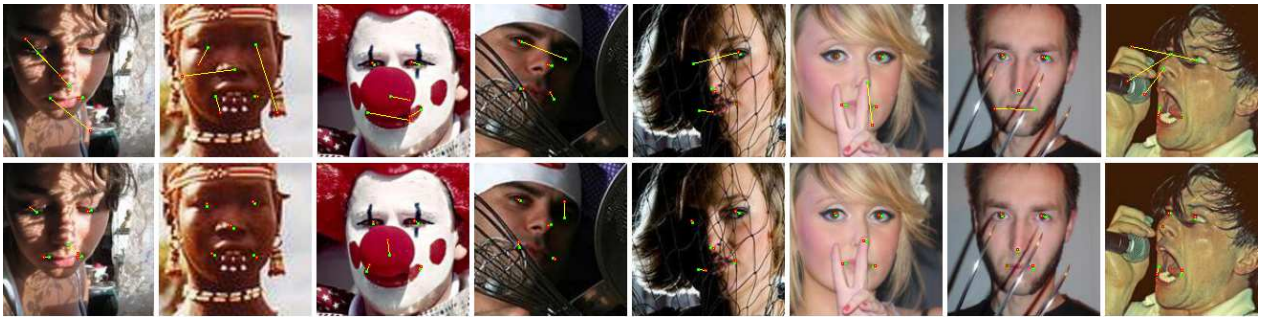

Figure 5. Samples from AFLW and AFW test sets showing keypoint detection accuracy without (top row) and with (bottom row) occlusion pre-processing using RCN.

Figure 6. Samples from TCDCN [41] (yellow border with green predicted points) versus SumNet (orange border) and RCN (blue border). In the latter two models, red and green dots show predicted and true keypoints. TCDCN samples are taken directly from [41].

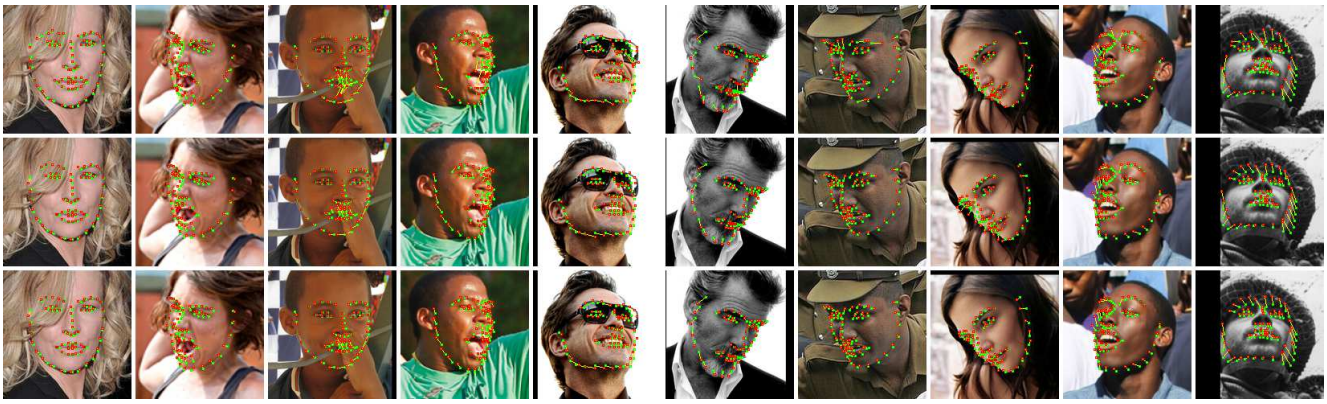

Figure 7. Samples from 300W test sets. Each column shows samples in this order (top to bottom): RCN, keypoint denoising model and the joint model. The first two columns show extreme expression and occlusion samples where RCN's prediction is highly accurate. The next 5 columns show samples where the denoising model improves the RCN's predictions. In the 8th column the structured model find a reasonable keypoint distribution but deteriorates the RCN's predictions. Finally, the last two columns show cases where the denoising model generates plausible keypoint distributions but far from the true keypoints.

References

[1] A. Asthana, T. Marks, M. Jones, K. Tieu, and M. Rohith. Fully automatic pose-invariant face recognition via 3d pose normalization. In *ICCV*, pages 937–944, 2011.

[2] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic. Robust discriminative response map fitting with constrained local models. In *CVPR*, pages 3444–3451, 2013.

[3] T. Baltrušaitis, P. Robinson, and L.-P. Morency. Continuous conditional neural fields for structured regression. In *ECCV*, pages 593–608. 2014.

[4] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio. Theano: new features and speed improvements. In *NIPS Workshop on Deep Learning*, 2012.

[5] X. Burgos-Artizzu, P. Perona, and P. Dollár. Robust face landmark estimation under occlusion. In *ICCV*, pages 1513–1520, 2013.

[6] X. Cao, Y. Wei, F. Wen, and J. Sun. Face alignment by explicit shape regression. In *ICCV*, 107(2):177–190, 2014.

[7] X. Cao, Y. Wei, F. Wen, and J. Sun. Face alignment by explicit shape regression. In *IJCV*, 107(2):177–190, 2014.

[8] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. In *PAMI*, 23(6):681–685, 2001.

[9] T. Cootes, C. Taylor, D. Cooper, and J. Graham. Active shape models-their training and application. In *CVIU*, 61(1):38–59, 1995.

[10] D. Cristinacce and T. Cootes. Feature detection and tracking with constrained local models. In *BMVC*, volume 2, page 6, 2006.

[11] D. Cristinacce and T. Cootes. Automatic feature localisation with constrained local models. *Pattern Recognition*, 41(10):3054–3067, 2008.

[12] D. Cristinacce and T. F. Cootes. Boosted regression active shape models. In *BMVC*, pages 1–10, 2007.

[13] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

[14] G. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, 07-49, University of Massachusetts, Amherst, 2007.

[15] M. Kostinger, P. Wohlhart, P. Roth, and H. Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *ICCV Workshop*, pages 2144–2151, 2011.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.

[17] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[18] S. Ren, X. Cao, Y. Wei, and J. Sun. Face alignment at 3000 fps via regressing local binary features. In *CVPR*, pages 1685–1692, 2014.

[19] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*, pages 234–241. Springer, 2015.

[20] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *ICCV Workshop*, pages 397–403, 2013.

[21] J. M. Saragih, S. Lucey, and J. F. Cohn. Face alignment through subspace constrained mean-shifts. In *ICCV*, pages 1034–1041, 2009.

[22] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *CVPR*, pages 3626–3633, 2013.

[23] P. Sermanet and Y. LeCun. Traffic sign recognition with multi-scale convolutional networks. In *IJCNN*, pages 2809–2813, 2011.

[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CVPR*, 2014.

[25] B. Smith and L. Zhang. Collaborative facial landmark localization for transferring annotations across datasets. In *ECCV*, pages 78–93. 2014.

[26] B. M. Smith, J. Brandt, Z. Lin, and L. Zhang. Nonparametric context modeling of local appearance for pose-and expression-robust facial landmark localization. In *CVPR*, pages 1741–1748, 2014.

[27] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *NIPS*, pages 1988–1996, 2014.

[28] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *CVPR*, pages 3476–3483, 2013.

[29] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2014.

[30] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, pages 1701–1708, 2014.

[31] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *CVPR*, 2015.

[32] G. Tzimiropoulos. Project-out cascaded regression with an application to face alignment. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 3659–3667. IEEE, 2015.

[33] G. Tzimiropoulos and M. Pantic. Gauss-newton deformable part models for face alignment in-the-wild. In *CVPR*, pages 1851–1858, 2014.

[34] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103, 2008.

[35] X. Xiong and F. De la Torre. Supervised descent method and its applications to face alignment. In *CVPR*, pages 532–539, 2013.

[36] H. Yang and I. Patras. Sieving regression forest votes for facial feature detection in the wild. In *ICCV*. IEEE, 2013.

[37] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. In *ICML Workshop on Deep Learning*, 2015.

[38] X. Yu, J. Huang, S. Zhang, W. Yan, and D. Metaxas. Pose-free facial landmark fitting via optimized part mixtures and

cascaded deformable shape model. In *ICCV*, pages 1944–1951, 2013.

[39] X. Yu, J. Huang, S. Zhang, W. Yan, and D. N. Metaxas. Pose-free facial landmark fitting via optimized part mixtures and cascaded deformable shape model. In *ICCV*, pages 1944–1951, 2013.

[40] J. Zhang, S. Shan, M. Kan, and X. Chen. Coarse-to-fine auto-encoder networks (cfan) for real-time face alignment. In *ECCV*, pages 1–16. 2014.

[41] Z. Zhang, P. Luo, C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *ECCV*, pages 94–108. 2014.

[42] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Learning deep representation for face alignment with auxiliary attributes. In *PAMI*, 2015.

[43] X. Zhao, S. Shan, X. Chai, and X. Chen. Locality-constrained active appearance model. In *ACCV*, pages 636–647. 2013.

[44] S. Zhu, C. Li, C. C. Loy, and X. Tang. Face alignment by coarse-to-fine shape searching. In *CVPR*, pages 4998–5006, 2015.

[45] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*, pages 2879–2886, 2012.