Feedback Networks
Supplementary Material

Amir R. Zamir[1,2*] Te-Lin Wu[1*] Lin Sun[1] William Shen[1] Jitendra Malik[2] Silvio Savarese[1]
[1] Stanford University [2] University of California, Berkeley
http://feedbacknet.stanford.edu/

Abstract

The following items are provided in the supplementary material:
1. A video clip describing the process.
2. Discussions on feedback via hidden state.
3. 'Feedback' vs 'Recurrent Feedforward'.
4. The Coarse-to-fine representation:
4.1. Timed-tSNE details.
4.2. Activation Maps of Feedback vs Feedforward.
5. Physical vs Virtual Depth.
6. Recurrent module choice: LSTM, GRU, RNN.
7. Stanford Cars dataset experimental details and analysis.
8. MPII dataset experimental details and analysis.

1. Video Clip

We provide a video clip in the supplementary material describing the pivotal aspects of the paper. To facilitate understanding the paper, we encourage watching the video available at https://youtu.be/MY5Uhv38Ttg.

2. Feedback via Hidden State

Feedback model predicts the output at each iteration and passes it to the next iteration. The common formulation of feedback is to explicitly feed back the thus-far output as part of next iteration's input (fig. 1, a). However, in our model, we pass this information via the hidden state that carries a direct notation of output rather than the explicit output; the output can be computed given the hidden state, using Eq. 4 of the main paper. This approach is illustrated in fig. 1, b, where f_i and f_o are learned functions relating the input i and output o to the hidden state.

Passing the output via a hidden state has two main advantages: first, as the input (e.g. images) and output (e.g. classes) belong to difference spaces, it enables developing generic feedback architectures without needing to design task-specific output-to-input functions [3]. Second, it brings further flexibility on the architecture and allows distributing the passed hidden states across different physical depths. In other words, there does not have to be a single hidden state relating the input to the output, and instead, there can be multiple hidden states each curated for a certain physical depth (i.e. Stack-All vs Stack-n architectures. See Table 2 of the main paper). This is shown in fig. 1, b and c.

However, it is not merely the recurrent structure that creates the feedback mechanism. In the next section, we show the conventional recurrent networks are mostly indeed feedforward and how our feedback model performs something rather different.

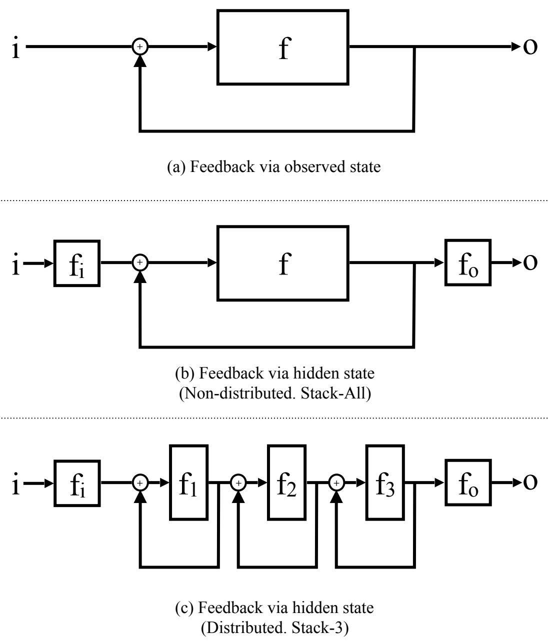
i
f
o
(a) Feedback via observed state
i
f_i
f
f_o
o
(b) Feedback via hidden state
(Non-distributed. Stack-All)
i
f_i
f_1
f_2
f_3
f_o
o
(c) Feedback via hidden state
(Distributed. Stack-3)

Figure 1. **Feedback in observed vs hidden state.** (a) The conventional feedback formulation, where the observed output is fed back in the input. (b) Feedback via the hidden state. No distribution of hidden state happens in this configuration (Stack-All architecture). (c) Feedback via the hidden state, where the hidden states are distributed (Stack-n architecture.)

*Both authors contributed equally.

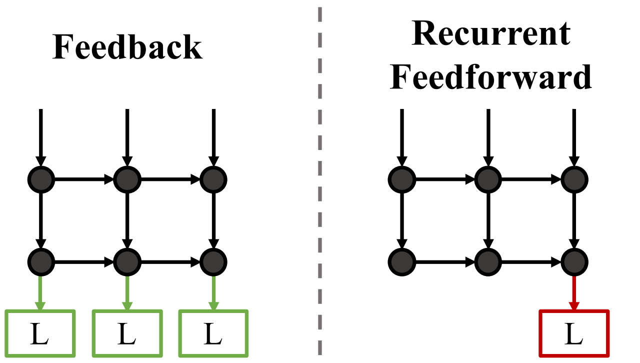

Figure 2. **Different ways of connecting the loss to a recurrent network, leading to instantiating a 'Feedback' or 'Recurrent Feedforward' model.**

3. 'Feedback' vs 'Recurrent Feedforward'

As explained in Sec. 3 of the main paper, feedback has two main requirements: 1) recurrence, and 2) rerouting a notion of posterior (output) into the system in each iteration. Employing a recurrent method for learning without fulfilling the second requirement leads to a 'recurrent feedforward' model [5]. Architecturally, the primary difference between feedback and recurrent feedforward is how the loss is connected. Fig. 2-right corresponds to the recurrent feedforward architecture where the loss is connected to the last iteration only. In this setting, the training phase is indeed a shared-weights feedforward operation when rolled out in time. As this model is only required to make one final prediction, it has the leisure to form the representation in a manner similar to feedforward but through the recurrent iterations, rather than physical layers. This hypothesis aligns with the comparison we made between the feedback model and its counterparts in the main paper's Sec. 4, especially Table 4. However, if the loss is connected to each iteration as in our feedback model (Fig. 2-left), the network is forced to make a prediction at each iteration and the hidden state will carry the thus-far output per Eq. 4 of the main paper. Therefore, instead of having the leisure to use several iterations to tackle the task, the network has to tackle the entire task at every iteration with support from the information passed down from the previous iteration, leading to a proper feedback model.

4. Representation in the Feedback Network

This sections provides a discussion on the representation developed by the feedback network. We use a variant of tSNE, timed-tSNE, to inspect how the representation evolves through the network when viewed from the window of final classification results. we also compare activation maps of feedback and feedforward models demonstrating that the two models develop significantly different representations, and therefore, different approaches to solving the problem though their endpoint numerical results are similar.

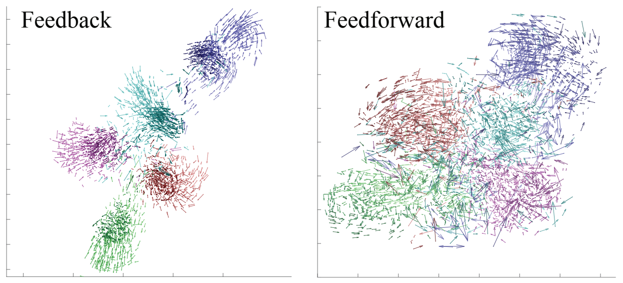

Figure 3. **Timed-tSNE** plots for five random CIFAR100 classes showing how the representation evolves through depth/iterations for each method (i.e. how a datapoint moved in representation space). The brighter the arrow, the earlier the depth/iteration. Feedback's representation is relatively disentangled throughout, while feedforward's representation gets disentangled only towards the end. (Best see on screen. Vector lengths are shown in half to avoid cluttering.)

4.1. Timed-tSNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) [6] is a visualization method for embedding high dimensional features in a lower dimensional space. We develop a variant of this method, called *Timed-tSNE*, which illustrates how the representation of a network evolves throughout depth/iterations, when viewed through the window of class labels. Instead of having one embedding location per datapoint (which is what original tSNE does), we form a trajectory for each datapoint by connecting a set of embedding locations. The k^{th} embedding location in a trajectory is the tSNE location of the corresponding datapoint using its representation at depth k while being intialized at its tSNE location at depth $k-1$. For the feedback network, the representations come from different iterations (i.e. i embeddings for a network with i iterations). For feedforward, the representations come from difference layers.

Fig. 3 shows the timed-tSNE plot for five random classes of CIFAR100; an animated version that better demonstrates the trajectories is included in the video. As apparent in Fig. 3, feedforward's trajectories are more intertwined than feedback's and only separate different classes at the last iterations. In contrast, feedback's timed-tSNE is more separated early on while evolving to targeted fine-grained margins. This aligns with the hypothesis that feedback network forms representations in a coarse-to-fine manner. It also aligns with the hypothesis for feedforward network that it forms representations in a low abstract to high abstract manner, which causes its earlier layer representations unsuitable for making final predictions.

4.2. Activations Maps

Fig. 4 provides neuron activations maps in the feedback, feedforward, and recurrent feedforward networks (see Sec. 3 for their distinctions) for a random CIFAR100 query

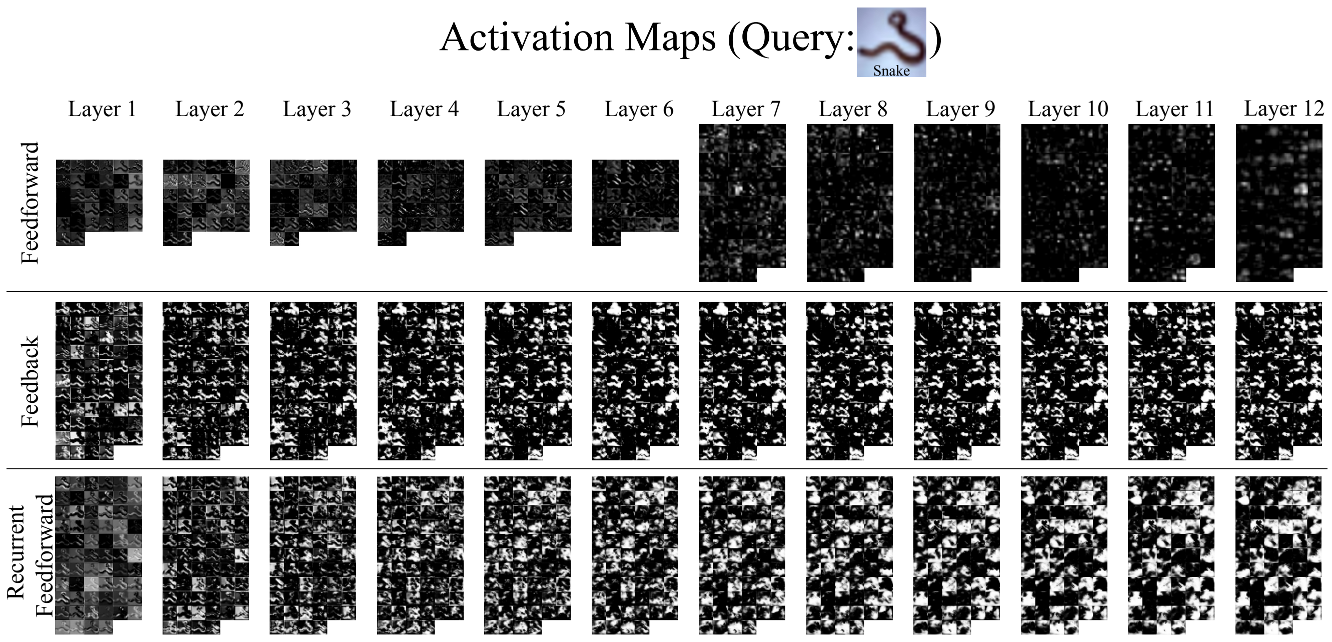

Figure 4. **Activation maps** of different layers/iterations of feedback, feedforward, and recurrent feedforward networks for a query image from CIFAR100. The feedback model develops a representation considerably different from feedforward.

image. The feedforward model develops the commonly observed edges-based activations in the early layers to sparse abstract activations in the late layers [8]. On the other hand, feedback network's activations show significantly dissimilar patterns suggesting a very different representation had been internally developed to solve the problem in hand. The activation of the feedback model appear to have a pattern consistent with a coarse-to-fine representation, as 1) early layers seem to have a notion of the object, unlike feedforward's early layers, 2) the activations are updated with rahter fine-grained changes as opposed to radical updates, and 3) low-level features, such as edges, are not observed in any layers. This observation is especially interesting since the endpoint performance of both feedback and feedforward models are close, suggesting that the networks took notably different routes with different properties for solving the problem, though they landed on the same performance in terms of endpoint results.

The activations of the recurrent feedforward model appear to be inbetween the feedforward and feedback models, suggesting that the weight-sharing mechanism is contributing to making the feedback's activations different from feedforward's (especially in terms of sparsity). However, it is apparent that recurrent feedforward's representation is also quite dissimilar to feedback's (see the edge-based activations in the early layers of recurrent feedforward and dissimilar and blobby activation patterns in the last layers) suggesting that the different representation of the feedback network is not entirely owed to the weight-sharing/recurrent mechanism, and the feedback is playing a role.

5. Physical vs Virtual Depth (optimal iteration number and physical depth)

In this section, we will discuss the optimal iteration number and physical depth for designing a feedback model. We observe that both iteration and physical depth positively correlate with performance, but naively adding iterations or substituting iterations with physical depth is detrimental to final performance.

5.1. Optimal Iteration Number

Fig. 5 provides the CAIFAR100 performance of two feedback models with two different physical depths (4 and 8) when *trained* for different number of iterations. We observe that for a fixed physical depth, there is a sweet spot for how many iterations one should train the network for. Among the iteration numbers that we experimented with, training for 8 iterations is optimal for physical depth 8 while training for 4 is optimal for physical depth 12. Another observation is that having iterations (in other words having feedback) is always better than not having them (iteration = 1). This experiment shows having feedback benefits a model but naively adding more iterations does not necessarily lead to a better final performance.

5.2. Relationship Between Physical Depth and Best Performance

We also observe that, fixing the total virtual depth, there is a sweet spot for the combination of physical depth and iteration number as multiple combinations can lead to the same virtual depth. Fig. 6 shows, for virtual depth 48, the optimal combination is physical depth 12 trained for 4 itera-

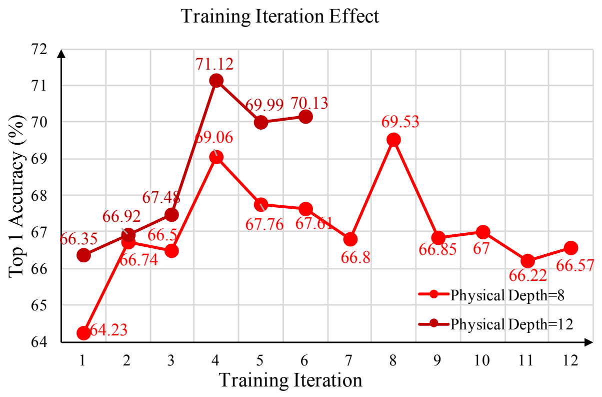

Figure 5. **Impact of number of training iterations on the endpoint performance on CIFAR100.** For two networks with physical depths 4 and 8, we train for different iteration and compare the results. The results suggest naively adding iterations does not necessarily lead to improving performance.

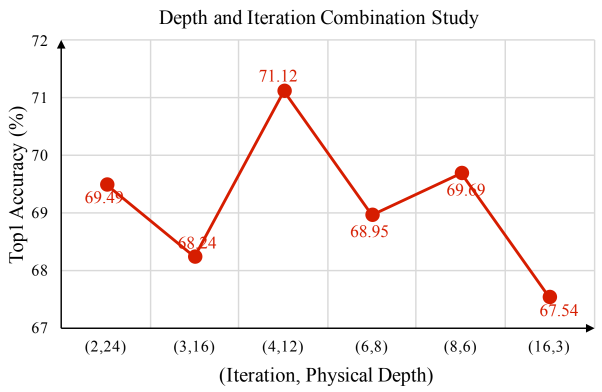

Figure 6. **CIFAR100 performance comparison of feedback models with same virtual depth but different (Iteration, Physical Depth) combinations.** All models have the virtual depth Iteration×Physical Depth=48.

tions. The fact that the (4,12) combination outperforms the combination with shallower physical depths shows *a certain minimum physical depth is required for a good performance*. In addition, (4,12) also outperforms combination with deeper physical depths showing that naively substituting iteration for deeper physical depth will hurt the final performance. Therefore, it is not beneficial to naively give up iteration for physical depth and vice versa, as both appear to be essential for achieving the best performance.

6. What to use as the feedback module?

As the most common recurrent model, we used LSTM to instantiate the feedback networks. However, in principle, any recurrent model can be used for this purpose. We performed a set of experiments to better understand this aspect.

6.1. LSTM vs GRU vs Vanilla RNN

We performed an experiment comparing ConvLSTM, ConvGRU and ConvRNN as the recurrent model adopted for instantiating feedback networks. LSTM acheives the best peformance followe by GRU and RNN (Table 1).

Network Type	Top1	Top5	Params (M)
ConvRNN	66.67	90.33	0.49
ConvGRU	68.38	91.22	1.10
ConvLSTM	71.11	91.48	1.90

Table 1. **Comparison of vanilla RNN, GRU, and LSTM as the recurrent module of feedback networks**, with physical depth = 12, iteration number = 4, virtual depth = 48.

6.2. LSTM Gate Ablation Study

To undertnad if/how the various internal mechanisms of LSTM fit the purpose of feedback based learning, we performed an ablation study of ConvLSTM gates on our 32 layer virtual depth feedback model. To ablate a gate in ConvLSTM, we substitute the convolutional structure of the gate (two layers deep of 3×3 convolutional filters) with a 1×1 convolutional layer. Table 2 provides the results showing that by disabling one or two gates of LSTMs, the performance drops by only 0.38% and 1.24% respectively while reducing the number of parameters to nearly three quarters or half. This suggests, though LSTM is currently the best recurrent model for instantiating feedback based learning, it is mostly likely not the optimal fit and future work could include developing recurrent models specifically designed for the purpose of feedback networks.

	Gate				Top1	Params
	input	forget	output	cell	%	(M)
Ablated 3	√	×	×	×	65.07	0.41
	×	√	×	×	63.13	
	×	×	√	×	65.76	
	×	×	×	√	65.61	
Ablated 2	√	√	×	×	67.21	0.73
	√	×	√	×	67.82	
	√	×	×	√	67.70	
	×	√	√	×	67.74	
	×	√	×	√	67.34	
	×	×	√	√	67.45	
Ablated 1	√	√	√	×	68.68	1.06
	√	√	×	√	67.28	
	√	×	√	√	68.34	
	×	√	√	√	68.40	
	√	√	√	√	69.06	1.39

Table 2. Performances comparison of adopting LSTM as the feedback module with different gate configuration.

Table 2 also shows among the four gates of ConvLSTM (input, output, forget and cell), the output gate is the most important as ablating it results in the largest performance drop. Also ablating all other gates except output will have

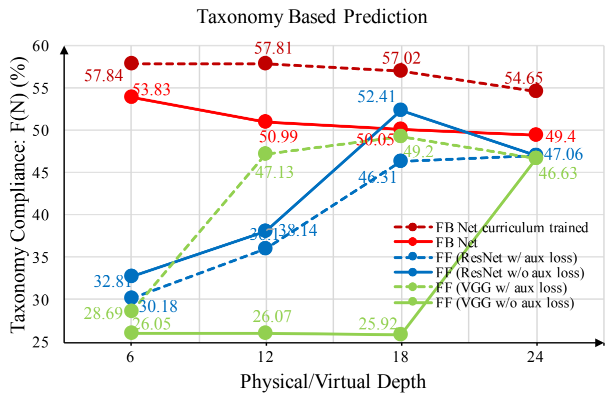

Figure 7. **Evalaution of taxonomoy based prediction for feedback (FB) and feedforward (FF) networks trained with or without auxiliary layers.** We use only fine loss to train, except for the curriculum learned one.

the highest performance among structures that have only one gate not ablated.

7. Stanford Cars Benchmark

The Stanford Cars dataset [4] contains 16,185 images (8,144 training, 8041 testing) of 196 fine classes and 7 coarse classes. The fine class labels describe Make, Model, Year of the car, e.g., "Tesla Model S Sedan 2012" or "Audi S5 Coupe 2012." The coarse class labels (Sedan, SUV, Coupe, Convertible, Pickup, Hatchback, Wagon) describe the basic type of the car. All training and testing images are resized to 96 $\times$ 96 during our training and testing phases. We perform the evaluations using the standard protocol of the dataset, the fine accuracy is the top1 accuracy for the 196 fine classes, while the coarse accuracy is the top1 accuracy for the 7 super classes.

We adopt a shallower network design to fit the smaller data volume of the dataset. The feedforward baseline model has the following architecture (naming convention as in main paper):

$\rightarrow C(3,16,3,1) \rightarrow BR$
$\rightarrow C(16,32,3,2) \rightarrow BR \rightarrow \{C(32,32,3,1) \rightarrow BR\}^5$
$\rightarrow C(32,64,3,2) \rightarrow BR \rightarrow \{C(64,64,3,1) \rightarrow BR\}^5$
$\rightarrow C(64,128,3,2) \rightarrow BR \rightarrow \{C(128,128,3,1) \rightarrow BR\}^5$
$\rightarrow Avg(12,1) \rightarrow FC(128,196)$

For feedback model we have virtual depth 24:

$\rightarrow C(3,16,3,1) \rightarrow BR$
$\rightarrow Iterate(16,32,3,2,2,4)$
$\rightarrow Iterate(32,64,3,2,2,4)$
$\rightarrow Iterate(64,128,3,1,2,4)$
$\rightarrow Avg(12,1) \rightarrow FC(128,196)$

In the next two subsections, we will demonstrates that the trends observed on CIFAR100 and reported in the main paper are also observed in the Stanford Cars dataset.

7.1. Taxonomy based Prediction Results

The resulting taxonomy based prediction performance is illustrated in Fig. 7. The same phenomenon as in CIFAR100 is observed: feedback network can achieve a good ability of making taxonomy based prediction even at the first iteration (or in other words, shallower virtual depth), while the feedforward networks do not achieve the same level of taxonomic prediction capacity until the later (deeper) layers.

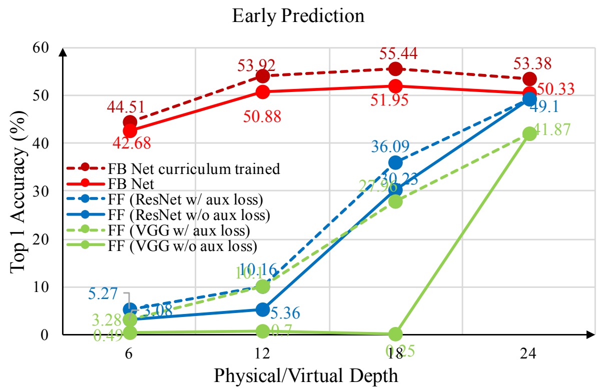

Figure 8. **Evaluation of early predictions.** Comparison of accuracy at the same virtual depths between feedback (FB) model and feedforward (FF) baselines (VGG & ResNet, with or without auxiliary loss layers).

7.2. Early Prediction Results

We demonstrate the early prediction advantage of the feedback network on the Stanford Cars dataset as well. We conduct a study at virtual depth 24 (similar results achieved with other depths) between the feedback and feedforward networks with various designs. As shown in Fig. 8, at the same virtual depth of 6, 12, and 18, the feedback network already achieves satisfactory and increasing accuracies. The feedforward baselines are designed in the same fashion as in main paper's early prediction section.

8. MPII Human Pose Estimation Dataset

We further study the effect of feedback on a regression task, MPII Human Pose Estimation [1]. We conduct this study on Hourglass [7], which is a feedforward structure significantly different from widely used feedforward models such as VGG or ResNet. This shows that feedback is not only applicable to conventional feedforward structures, but also more complex ones.

MPII Human Pose Estimation dataset [1] consists of 40k samples (28k training, 11k testing). Since the test annotations are not provided, we train on a subset of training images while evaluating on a held-out validation set of 3000 samples. We perform the evaluations using the standard protocol of the dataset by calculating PCKh value. PCKh (Percentage of Correct Keypoints-head) metric measures the percentage of detections that fall within a normalized distance (head size) of the ground truth.

We design our feedback model based on one-stack Hourglass [7]. As shown in Fig. 9, we replace ResNet bottle neck units (olive colored blocks) with ConvLSTM (green

Method	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	PCKh
Carreira et al. IEF [3]	95.7	91.7	81.7	72.4	82.8	73.2	66.4	81.3
Belagiannis&Zisserman, recurrent [2]	97.2	92.6	84.6	78.4	83.7	75.7	70.0	83.9
Ours	97.6	92.9	86.3	81.0	84.9	75.8	70.0	85.0

Table 3. Performance comparison of different feedback models on MPII human pose estimation.

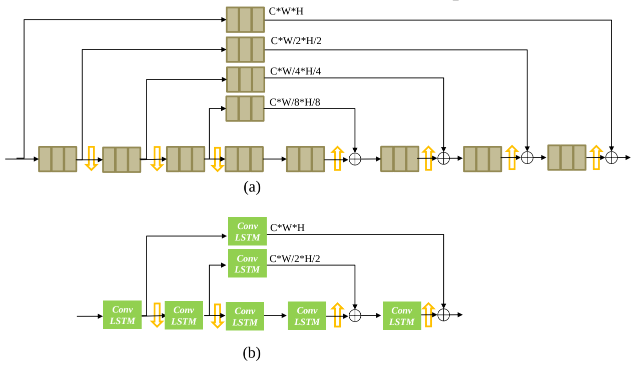

Figure 9. **Structural comparison between one stack feedforward hourglass (a) and one stack feedback hourglass (b).** ↑ denotes upscaling the resolution, and ↓ denotes downsizing the resolution. Both structures combine different scale of features

blocks). Each ConvLSTM unit has physical depth of 1 with iteration number equal to the number of ResNet blocks it substitutes. We also perform the same up (↑) and down (↓) sampling as hourglass. We only do up-and-down sampling twice (Hourglass does so four times), as even with fewer up-and-down sampling the feedback model outperforms the Hourglass baseline.

In Table 3, we compare our model with other feedback based methods [3, 2]. IEF [3] represents the conventional way of applying feedback (via observed state), while our model represents the feedback concept discussed in Sec.2. Qualitative results of our method along with IEF [3] and one stack feedforward hourglass are provided in Fig. 10.

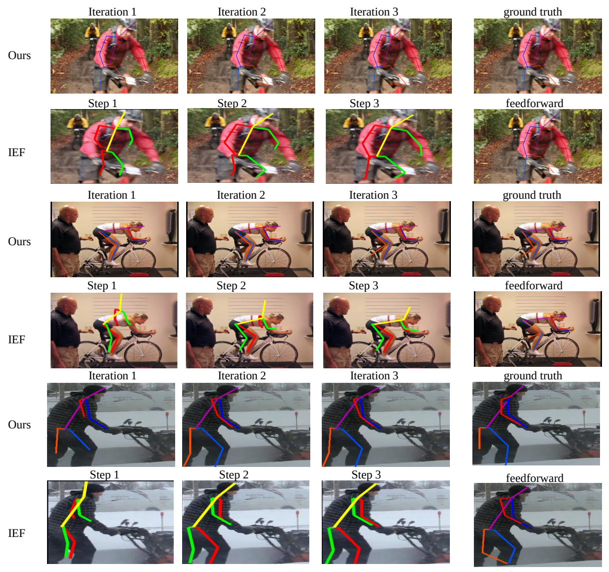

Figure 10. Qualitative results on MPII human pose estimation. From the left to right and top to bottom, our proposed method with three iterations and the ground truth as well as IEF with three steps and feedforward network results.

References

[1] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3686–3693. IEEE, 2014.

[2] V. Belagiannis and A. Zisserman. Recurrent human pose estimation. *arXiv preprint arXiv:1605.02914*, 2016.

[3] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. *arXiv preprint arXiv:1507.06550*, 2015.

[4] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013.

[5] M. Liang and X. Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3367–3375, 2015.

[6] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[7] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. *arXiv preprint arXiv:1603.06937*, 2016.

[8] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.