# Tekin, Márquez-Neila, Salzmann, Fua

Learning to Fuse 2D and 3D Image Cues for Monocular Body Pose Estimation

## 1 Summary

This paper introduces a trainable fusion scheme which **learns how to combine 2D joint heatmaps and image features** to produce a 3D pose estimate. Once we have the 2D joint heatmaps, we pass them and the image through another network which provides the option to combine image feature maps with heatmap feature maps at multiple/any stages and lets the network decide how, i.e. according to learned weights. At the end of the day, the fusion network outputs the 3D pose.

## 2 In one sentence fragment

new *3D network* which fuses *image features* and *2D heatmaps* around a variable layer of the network, where the weights at each layer for the linear combination of (a) *fusion stream* and (b) *decorrelated image and heatmap features* depend on trainable parameters

## 3 Fusion network

Fusion occurs "around" a specific layer $\beta$; the weight at layer $l$ is defined by the function

$$w_l = \frac{1}{1 + e^{-\alpha(l-\beta)}}$$

where $\alpha$ represents how sharp the transition from weights of 0 to weights of 1 is. (Weights will all be 0 in the first layers, representing that the streams are still split, and then after transitioning to 1 it means that the fusion has happened and we are continuing with an already-fused network.)

The loss is

$$L(\theta, \alpha, \beta) = \sum_{n=1}^{N} \|f(\mathbf{i}_n, \mathbf{X}_n; \theta, \alpha, \beta) - \mathbf{y}_n\|_2^2 + \frac{\lambda}{\alpha^2}$$

It is designed to penalize small values of $\alpha$, because we want a sharp transition (i.e. fusion occurring mostly at a single layer). $\alpha \to 0$ means "mix data and fusion streams equally at all layers" (what we *don't* want).

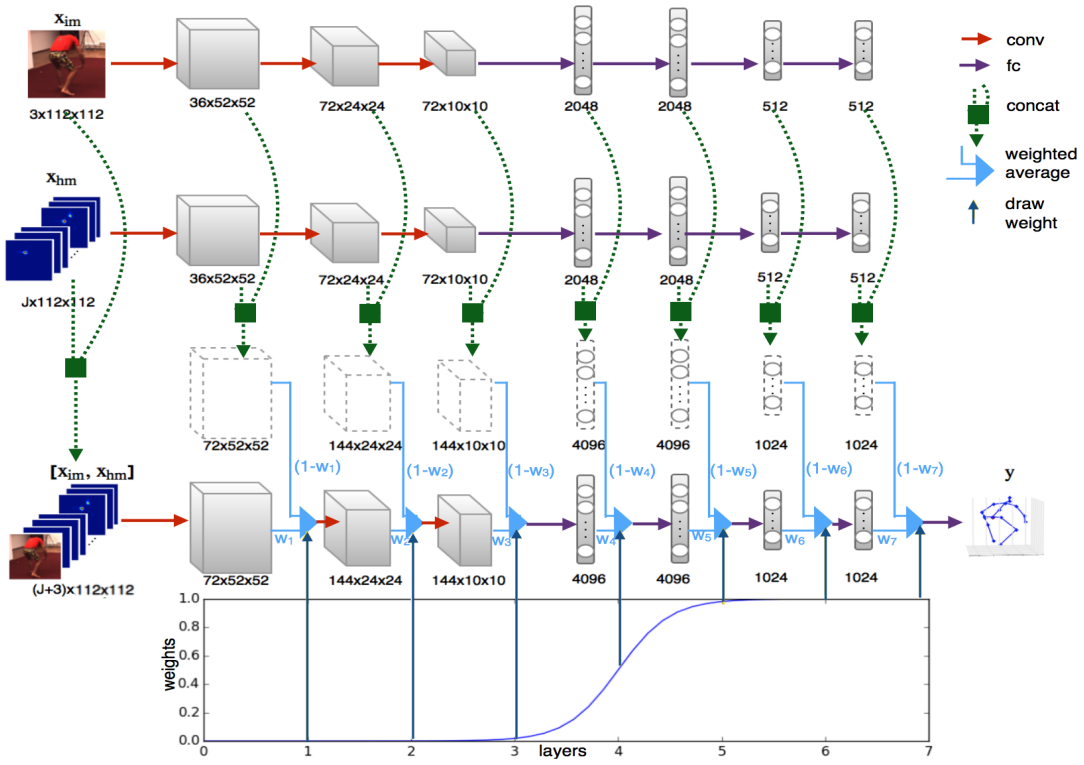
Since the weights (and thus ultimately the output) are defined according to the earlier sigmoid function of $\alpha$ and $\beta$, $\alpha$ and $\beta$ will both be learned during training.

## 4 The point

The network gets to take into account both *image cues* (which might help resolve 3D ambiguities) and *2D pose*. It is able to fuse this information in an adaptable (optimizable) manner.

## 5 Random notes

- We (allegedly) don't want to mix data and fusion streams at every layer because it'll be too large of a network with too many parameters (and this will be inefficient and prone to overfitting).




source: Tekin, Márquez-Neila, Salzmann, and Fua in [1]

# References


[1] Bugra Tekin, Pablo Márquez-Neila, Mathieu Salzmann, Pascal Fua. Learning to Fuse 2D and 3D Image Cues for Monocular Body Pose Estimation. *arXiv preprint arXiv:1611.05708 (2017).*


# Evaluation

- Evaluate on H36M (train on S1, S5, S6, S7, S8 / test on S9, S11) via Euclidean distance. (For comparison with certain papers' results, align estimated skeleton to ground-truth skeleton via Procrustes transformation.)
  - outperforms other methods (though none I'm familiar with) on each action by a significant margin
  - trainable fusion does better than all hardcoded strategies for fusion (that they attempted, at least)
- Also evaluate on **HumanEva-I** (benchmark for 3D human pose estimation), **KTH Multiview Football II** (outdoor soccer data), and **LSP** (in-the-wild 2D dataset, only evaluated 3D qualitatively here).
- Typically, fusion wants to happen later in the network, owing possibly to the greater discriminative power provided by long-running, uncorrelated *heatmap* and *image* streams.
- The $\lambda/\alpha^2$ regularization (encouraging sharp fusion) helps.

  $\rightarrow$ less error and more efficiency (since with sharp fusion, a lot of the network can be cut out)