Supplementary material: Multi-task head pose estimation in-the-wild

Roberto Valle, José M. Buenaposada and Luis Baumela

Abstract—We present the following supplementary material:
- Further details about the MNN architecture.
- Images and videos for qualitative evaluation.

1 MNN Architecture

Here, we present our Multi-task Neural Network (MNN) architecture in detail. It is inspired in similar encoder-decoder architectures such as U-Net [1], RCN [2] or Hourglass [3].

We show in Figs. 1 and 2 the MNN encoder-decoder with 9 stages, reducing the spatial extent of the input image from 256×256 to 1×1 pixels. It includes lateral skip connections that link symmetric layers between the encoder and the decoder, but we do not display them for the sake of simplicity.

We modify the original U-Net architecture introducing bottleneck residual blocks [4] instead of their original convolutional layers. The residual block lets us reduce the number of operations and increase depth while preserving the gradient back propagation through. Since the residual modules require that the input and output of each block have the same dimensions, we include in the decoder additional 1×1 convolutions to reduce the number of feature maps accordingly. Whenever the spatial resolution is halved we double the number of feature maps, from 64 and up to 256. We show in Tables 1 and 2 further details about our encoder and decoder respectively. We also added BatchNormalization and ReLu after each convolutional layer, but we do not display them for the sake of simplicity.

At the end of the encoder (*i.e.*, bottleneck layer) we hook up two different losses, $\mathcal{L}_{ED}(\mathbf{p}^{ED})$ and $\mathcal{L}_{AE}(\mathbf{p}^{AE})$ related to the head pose estimation, adding two 1×1 Conv2D layers with 6 feature maps (`fc_pose` and `fc_pose_proj`). The former directly minimizes the euclidean error of pose parameters. The latter measures the alignment error produced by projecting a mean 3D face model using these pose parameters.

At the end of the decoder we hook up two cross-entropy losses, $\mathcal{L}_{CE}(\mathbf{h})$ and $\mathcal{L}_{CE}(\mathbf{v})$, related to the estimation of non-rigid landmark location and their visibilities by adding two different 1×1 Conv2D layers with one feature map per landmark (`fc_lnd` and `fc_vis`). For the face alignment task, we introduce a softmax activation layer to provide probability distributions for the location of each landmark. For the visibility task, we add an AveragePooling layer to estimate each landmark visibility.

The MNN encoder-decoder has $4,900,948$ parameters for L=68, while the model size is 20.3 MB. During testing, we may dispose of the decoder to build a very efficient head pose estimation module with only $2,048,710$ parameters, reducing the model size to 8.5 MB.

2 Qualitative imagery and video results

The problem of estimating the head pose from a single RGB image acquired in the most challenging in-the-wild conditions is still far from being completely solved. In this section we discuss some of the worst estimations of our model. To this end, in Fig. 3 we have selected some representative error images where the average MAEs (yaw, pitch, roll) is greater than 5°. We show errors due to partial occlusions (*e.g.*, 4th image in Fig. 3a), extreme rotations (*e.g.*, 3rd image in Fig. 3b), exaggerated facial expressions (*e.g.*, 2nd image in Fig. 3c), presence of make-up (*e.g.*, 2nd image in Fig. 3d) or illumination (*e.g.*, 5th image in Fig. 3e). Also, we note that the head pose predictions in the 3rd image in Fig. 3d and 6th image in Fig. 3e are coherent with an

Name	Layer	Output	Connected to
`input`	InputLayer	(256, 256, 3)	
`conv_9_1`	Conv2D (1x1)	(256, 256, 64)	`input`
`conv_9_2`	Conv2D (1x1)	(256, 256, 64)	`conv_9_1`
`conv_9_3`	Conv2D (3x3)	(256, 256, 64)	`conv_9_2`
`conv_9_4`	Conv2D (1x1)	(256, 256, 64)	`conv_9_3`
`add_9_4`	Add	(256, 256, 64)	`conv_9_1`, `conv_9_4`
`conv_8_1`	Conv2D (2x2)	(128, 128, 128)	`conv_9_4`
`conv_8_2`	Conv2D (1x1)	(128, 128, 64)	`conv_8_1`
`conv_8_3`	Conv2D (3x3)	(128, 128, 64)	`conv_8_2`
`conv_8_4`	Conv2D (1x1)	(128, 128, 128)	`conv_8_3`
`add_8_4`	Add	(128, 128, 128)	`conv_8_1`, `conv_8_4`
`conv_7_1`	Conv2D (2x2)	(64, 64, 256)	`conv_8_4`
`conv_7_2`	Conv2D (1x1)	(64, 64, 64)	`conv_7_1`
`conv_7_3`	Conv2D (3x3)	(64, 64, 64)	`conv_7_2`
`conv_7_4`	Conv2D (1x1)	(64, 64, 256)	`conv_7_3`
`add_7_4`	Add	(64, 64, 256)	`conv_7_1`, `conv_7_4`
...	...	...	...
`conv_2_1`	Conv2D (2x2)	(2, 2, 256)	`conv_3_4`
`conv_2_2`	Conv2D (1x1)	(2, 2, 64)	`conv_2_1`
`conv_2_3`	Conv2D (3x3)	(2, 2, 64)	`conv_2_2`
`conv_2_4`	Conv2D (1x1)	(2, 2, 256)	`conv_2_3`
`add_2_4`	Add	(2, 2, 256)	`conv_2_1`, `conv_2_4`
`pool_1_1`	MaxPooling (2x2)	(1, 1, 256)	`conv_2_4`
`conv_1_2`	Conv2D (1x1)	(1, 1, 64)	`conv_1_1`
`conv_1_3`	Conv2D (3x3)	(1, 1, 64)	`conv_1_2`
`conv_1_4`	Conv2D (1x1)	(1, 1, 256)	`conv_1_3`
`add_1_4`	Add	(1, 1, 256)	`pool_1_1`, `conv_1_4`
`fc_pose`	Conv2D (1x1)	(1, 1, 6)	`add_1_4`
`fc_pose_proj`	Conv2D (1x1)	(1, 1, 6)	`add_1_4`

TABLE 1: MNN encoder architecture.

Name	Layer	Output	Connected to
`up_1_5`	UpSampling (2x2)	(2, 2, 256)	`add_1_4`
`concat_2_5`	Concatenate	(2, 2, 512)	`add_2_4`, `up_1_5`
`conv_2_6`	Conv2D (1x1)	(2, 2, 256)	`concat_2_5`
`conv_2_7`	Conv2D (1x1)	(2, 2, 64)	`conv_2_6`
`conv_2_8`	Conv2D (3x3)	(2, 2, 64)	`conv_2_7`
`conv_2_9`	Conv2D (1x1)	(2, 2, 256)	`conv_2_8`
`add_2_9`	Add	(2, 2, 256)	`conv_2_6`, `conv_2_9`
`conv_2_10`	Conv2DTrans (2x2)	(4, 4, 256)	`add_2_9`
...	...	...	...
`concat_7_5`	Concatenate	(64, 64, 512)	`add_7_4`, `conv_6_10`
`conv_7_6`	Conv2D (1x1)	(64, 64, 256)	`concat_7_5`
`conv_7_7`	Conv2D (1x1)	(64, 64, 64)	`conv_7_6`
`conv_7_8`	Conv2D (3x3)	(64, 64, 64)	`conv_7_7`
`conv_7_9`	Conv2D (1x1)	(64, 64, 256)	`conv_7_8`
`add_7_9`	Add	(64, 64, 256)	`conv_7_6`, `conv_7_9`
`conv_7_10`	Conv2DTrans (2x2)	(128, 128, 128)	`add_7_9`
`concat_8_5`	Concatenate	(128, 128, 256)	`add_8_4`, `conv_7_10`
`conv_8_6`	Conv2D (1x1)	(128, 128, 128)	`concat_8_5`
`conv_8_7`	Conv2D (1x1)	(128, 128, 64)	`conv_8_6`
`conv_8_8`	Conv2D (3x3)	(128, 128, 64)	`conv_8_7`
`conv_8_9`	Conv2D (1x1)	(128, 128, 128)	`conv_8_8`
`add_8_9`	Add	(128, 128, 128)	`conv_8_6`, `conv_8_9`
`conv_8_10`	Conv2DTrans (2x2)	(256, 256, 64)	`add_8_9`
`concat_9_5`	Concatenate	(256, 256, 128)	`add_9_4`, `conv_8_10`
`conv_9_6`	Conv2D (1x1)	(256, 256, 64)	`concat_9_5`
`conv_9_7`	Conv2D (1x1)	(256, 256, 64)	`conv_9_6`
`conv_9_8`	Conv2D (3x3)	(256, 256, 64)	`conv_9_7`
`conv_9_9`	Conv2D (1x1)	(256, 256, 64)	`conv_9_8`
`add_9_9`	Add	(256, 256, 64)	`conv_9_6`, `conv_9_9`
`fc_lnd`	Conv2D (1x1)	(256, 256, L)	`add_9_9`
`fc_vis`	Conv2D (1x1)	(256, 256, L)	`add_9_9`
`pool_vis`	AvgPooling (256x256)	(1, 1, L)	`fc_vis`

TABLE 2: MNN decoder architecture.

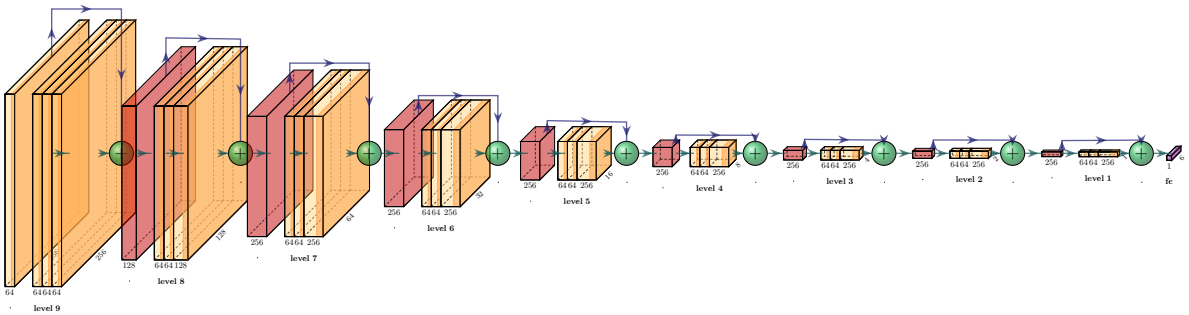


Fig. 1: MNN encoder diagram. We locate at the end of the encoder the losses that minimize the head pose, $\mathcal{L}_{ED}(\mathbf{p}^{ED})$, and the rigid landmarks location errors, $\mathcal{L}_{AE}(\mathbf{p}^{AE})$. For each task, we use a fully connected layer with 6 outputs (yaw, $pitch$, $roll$, t_x, t_y, t_z).

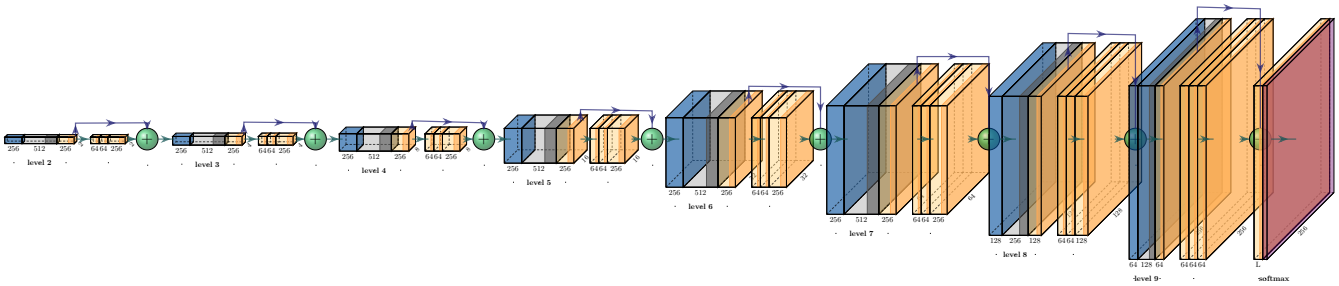


Fig. 2: MNN decoder diagram. We locate at the end of the decoder the losses that minimize the landmarks location error, $\mathcal{L}_{CE}(\mathbf{h})$, and their visibility estimation, $\mathcal{L}_{CE}(\mathbf{v})$. For the face alignment task, we use a last softmax activation layer to generate heatmaps from the $[256 \times 256 \times L]$ layer, where L represents the number of landmarks. For the visibility task, we use a pooling layer with kernel size 256×256.

incorrect face landmark detection, which denotes that our head pose estimations are usually linked with the face alignment task.

We also provide qualitative results obtained by processing two videos `mnn.mp4` and `mnn+or.mp4` from Biwi [5] and 300VW [6] data sets respectively. We process each frame individually without tracking or filtering whatsoever between frames. Here, we use the model trained using 300W-LP [7] for both videos, where we pre-train with landmarks and refine for all three tasks (*i.e.*, head pose, face landmark location and visibility estimation).

In `mnn.mp4` (see Fig. 4) we use our encoder-only model to process a sequence including frontal and profile face orientations. For this video we obtain 2.23 MAE for yaw, 3.94 MAE for pitch and 2.99 MAE for roll, which represent on average a 3.05 MAE for all three angles. The performance of our model in this sequence is close to the average in Biwi, 3.66 (see Table 2 in the paper). Observe that yaw and pitch angles are typically the most difficult to estimate, perhaps because both produce the largest appearance changes in the expressive parts of the face.

In `mnn+or.mp4` (see Fig. 1 in the paper) we can appreciate not only the robustness in the estimation of self-occlusions, but also the remarkable accuracy and stability achieved in the computation of face landmark locations.

REFERENCES

[1] 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, pp. 234–241.

[2] S. Honari, J. Yosinski, P. Vincent, and C. J. Pal, "Recombinator networks: Learning coarse-to-fine feature aggregation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5743–5752.

[3] J. Yang, Q. Liu, and K. Zhang, "Stacked hourglass network for robust facial landmark localisation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 2025–2033.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. European Conference on Computer Vision*, 2016, pp. 630–645.

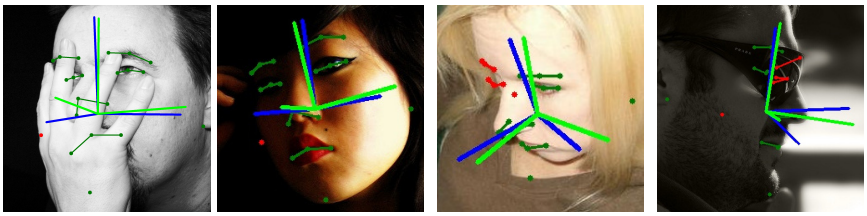
[5] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool, "Random forests for real time 3D face analysis," *International Journal of Computer Vision*, vol. 101, no. 3, pp. 437–458, 2013.

[6] J. Shen, S. Zafeiriou, G. G. Chrysos, J. Kossaifi, G. Tzimiropoulos, and M. Pantic, "The first facial landmark tracking in-the-wild challenge: Benchmark and results," in *Proc. International Conference on Computer Vision Workshops*, 2015, pp. 1003–1011.

[7] X. Zhu, X. Liu, Z. Lei, and S. Z. Li, "Face alignment in full pose range: A 3D total solution," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 41, no. 1, pp. 78–92, 2017.

[11.17, -7.48, 16.18] [-4.63, -20.21, -17.75] [8.73, 4.91, -17.67] [-7.48, -3.85, -16.82] [10.48, 14.73, -0.27] [-2.15, 3.25, -5.52]
[0.67, -10.77, 12.62] [-8.86, -7.89, -16.42] [13.11, -0.44, -10.59] [-2.21, 2.06, -9.24] [-0.63, 6.19, 9.32] [-6.90, 0.60, 14.10]

(a) COFW



[-31.99, -4.89, 0.48] [-42.31, -3.04, -8.32] [-73.73, -28.84, 25.15] [67.69, 2.13, 21.22] [2.71, 3.07, 12.46] [10.14, -16.72, -83.27]
[-26.29, 9.61, -4.19] [-21.76, 0.86, -14.86] [-48.65, -30.74, 18.76] [80.21, -8.37, 5.75] [11.12, 13.99, 24.42] [2.11, -22.64, -81.79]

(b) AFLW

[-0.67, 6.83, 4.27] [6.46, -14.16, -24.24] [6.23, 42.95, 2.72] [13.23, 44.29, -26.65] [-16.09, 5.12, -1.66] [-6.67, -11.14, -1.99]
[6.58, -0.83, 1.34] [-1.37, -8.79, -21.10] [5.06, 32.48, -2.49] [13.76, 28.97, -24.74] [-33.02, 7.67, 5.45] [-19.75, -2.81, 16.87]

(c) 300W (d) WFLW

[18.58, 22.27, 0.78] [-32.11, 36.13, -30.79] [52.41, -50.13, 5.14] [-48.95, -6.10, 0.39] [-9.50, -14.64, 2.03] [23.29, 3.88, -101.91]
[29.39, 18.03, 2.08] [-26.60, 30.18, -19.69] [32.73, -37.82, 18.09] [-56.35, -9.25, 7.87] [-29.05, -11.67, 8.41] [36.63, 10.42, 4.88]

(e) AFLW2000-3D

Fig. 3: Representative faces with large prediction errors in COFW, AFLW, 300W, WFLW and AFLW2000-3D testing data sets. The co-ordinate systems and [yaw, pitch, roll] angles below represent ground truth and predicted head pose using blue and green colours. Moreover, green and red face landmarks show visible and occluded predictions respectively.

Fig. 4: Some processed frames from a Biwi sequence in laboratory conditions (see `mnn.mp4`). In blue and green we display respectively Biwi annotations and MNN predictions. Yaw, pitch and roll angles are also shown enclosed in brackets.