ERA: Enhanced Rational Activations

Martin Trimmel*[1][0000−0001−5991−9845], Mihai Zanfir*[2][0000−0001−9347−0738], Richard Hartley[2,3][0000−0002−5005−0191], and Cristian Sminchisescu[1,2][0000−0001−5256−886X]
[1]Lund University [2]Google Research [3]Australian National University
martin.trimmel@math.lth.se
{mihaiz, richardhartley, sminchisescu}@google.com

In this supplementary material, we provide additional details and experiments for *ERA* on both image classification, and 3d human pose and shape reconstruction tasks. For the former, we report results also on the MNIST and ImageNet datasets and we show more ablation studies on the CIFAR10 dataset, together with more examples of learned *ERA*s and plots of loss surfaces. For the latter task, we provide more details for training and ablate some architectural choices.

A Image Classification

Implementation details. We train neural networks to perform image classification on the MNIST [9], CIFAR10 [8] and ImageNet [3] datasets, using the Tensorflow [1] framework. As in the main part of the paper, we focus on two small architectures on MNIST and CIFAR10. We consider fully-connected (FCN) and convolutional neural (CNN) networks, defined recursively by

$$\mathbf{x}_{i+1} = (\sigma_i \circ \ell_i)(\mathbf{x}_i), \quad 1 \leq i \leq 4; \tag{1}$$

where $\mathbf{x}_1$ is the input, ℓ_i is the i^{th} linear layer of the network, followed by an activation function σ_i. In the fully-connected case, ℓ_i is a dense layer and in the convolutional case, ℓ_i is a convolutional layer. The CNN consists of 2 convolutional layers with 64 filters, followed by 2 dense layers with 128 and 10 nodes, respectively. The fully-connected network consists of 4 dense layers with 512, 256, 128 and 10 nodes, respectively. The fully-connected and the convolutional neural network have $\sim 1,740,000$ and $\sim 2,140,000$ parameters, respectively. On ImageNet, we use a ResNet-18 with $\sim 340,000$ parameters. As optimizer we use SGD with momentum (with momentum parameter 0.9.) On CIFAR10, we train the networks for 100 epochs using a batch size of 64 and an initial learning rate of 0.01 which decays by a factor of 10 after having completed 25, 50 and 75 training epochs. On MNIST, we train the networks for 100 epochs using a batch size of 32 and an initial learning rate of 10^{-4} which decays by a factor of 10 after having completed 10, 20, 30, 40, 50 and 60 epochs. On ImageNet, we train the networks for 200 epochs using a batch size of 1,024 and an initial learning rate of 10^{-3} which decays by a factor of 10 after 80, 120, 160 and 180 epochs. All

* Denotes equal contribution.

non-rational network layers use l2 weight regularization with parameter 0.001. We apply data augmentation with horizontal and vertical shifts of up to 6 pixels and enabling horizontal flipping of images.

A.1 Additional Results on CIFAR10

Batch Normalization Training a CNN with the same training configuration as in Table 2 with the normalization replaced by BatchNorm, we observe the following test accuracies (over 5 runs): Leaky ReLU: 69.4±0.4%; ReLU: 72.2±0.2%; Swish: $72.2 \pm 0.4\%$; ERA: $74.4 \pm 0.3\%$. Intuitively, we want to keep the distribution of the inputs to the activation fixed during training because this makes learning for rational activations easier. When using BatchNorm, the preactivation distribution of a given data point is influenced by statistics from all the other data points in the batch, which could make learning more difficult. This may be the reason why ERA achieves larger gains over the baselines when instance normalization is used instead of batch normalization.

Experiments with MobileNet When running the official TensorFlow implementation of MobileNet [4] with BatchNorm with ReLU (ReLU-6 variant as originally proposed) activations on CIFAR10, we get 78.3±0.2% and 69.9±2.0% test accuracy when using the Adam and SGD optimizers, respectively. When we replace the ReLUs with ERAs, the test accuracies increase $78.9 \pm 1.3\%$ and $77.4 \pm 0.6\%$, respectively.

Plots of *ERA*s Figures 1 and 2 show plots of the *ERA*s corresponding to networks shown in Figure 1 and 2 from the main part of the paper, respectively. In line with the results in the main paper, we observe that the learned *ERA*s differ drastically from the initializations. As noted previously, compared to Swish initialisation, random initialisation can result in very small or large derivative values which can make the training more difficult. Moreover, the learned functions in different layers seem to be more similar for Swish initialisation, whereas they differ drastically for random initialisation.
We re-trained networks with the same configurations used in Figure 1 and 2 from the main part of the paper. The plots of the learned *ERA*s are shown in Figures 3, 4, 5 and 6. We observe that rerunning the same configuration can produce very different activation functions, in particular for randomly initialised activations.

Loss Landscapes Figure 7 shows two additional loss landscapes of networks trained on CIFAR10. As for the loss landscapes in the main paper, the network at the bottom converged, whereas the network at the top did not converge. This might be due to multiple local minima visible in the the blue part of the plots in row 1. It is clear from the plot in row 2 that for a bad choice of initialisation, the network risks getting stuck in a local minimum. This underlines the benefit of Swish initialisation over random initialisation.

Effect of Gradient Clipping Table 1 shows the effect of gradient clipping on the performance of the network after training. Gradient clipping greatly stabilises the training of randomly initialised *ERA*s, but it does not seem to offer a

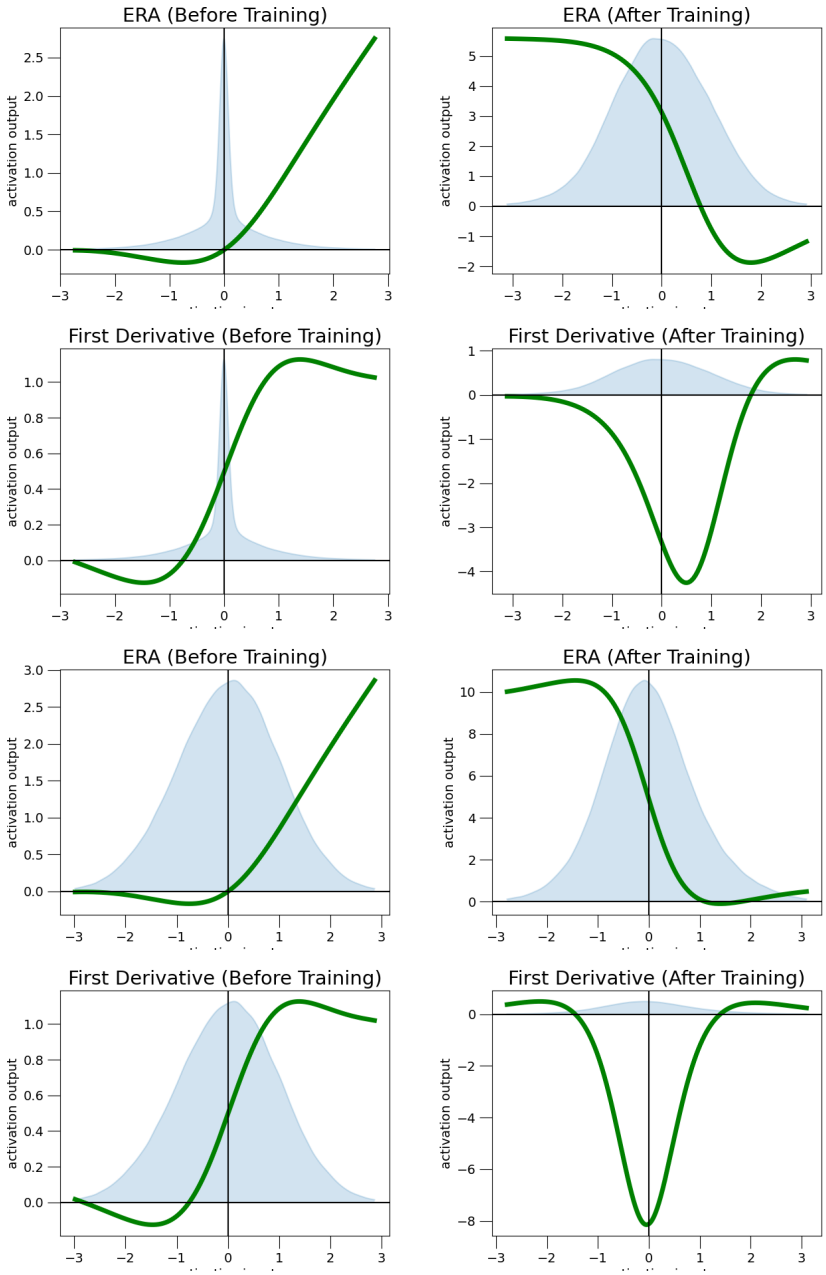

Fig. 1: Plots of the second (row 1 & 2) and third (row 3 & 4) *ERA* of the same CNN as in Figure 1 in the main paper. The *ERA* is initialised as a Swish activation and has degree 5/4. The shaded curve in the background shows the density distribution of the input to the activation (not true to scale on y-axis).

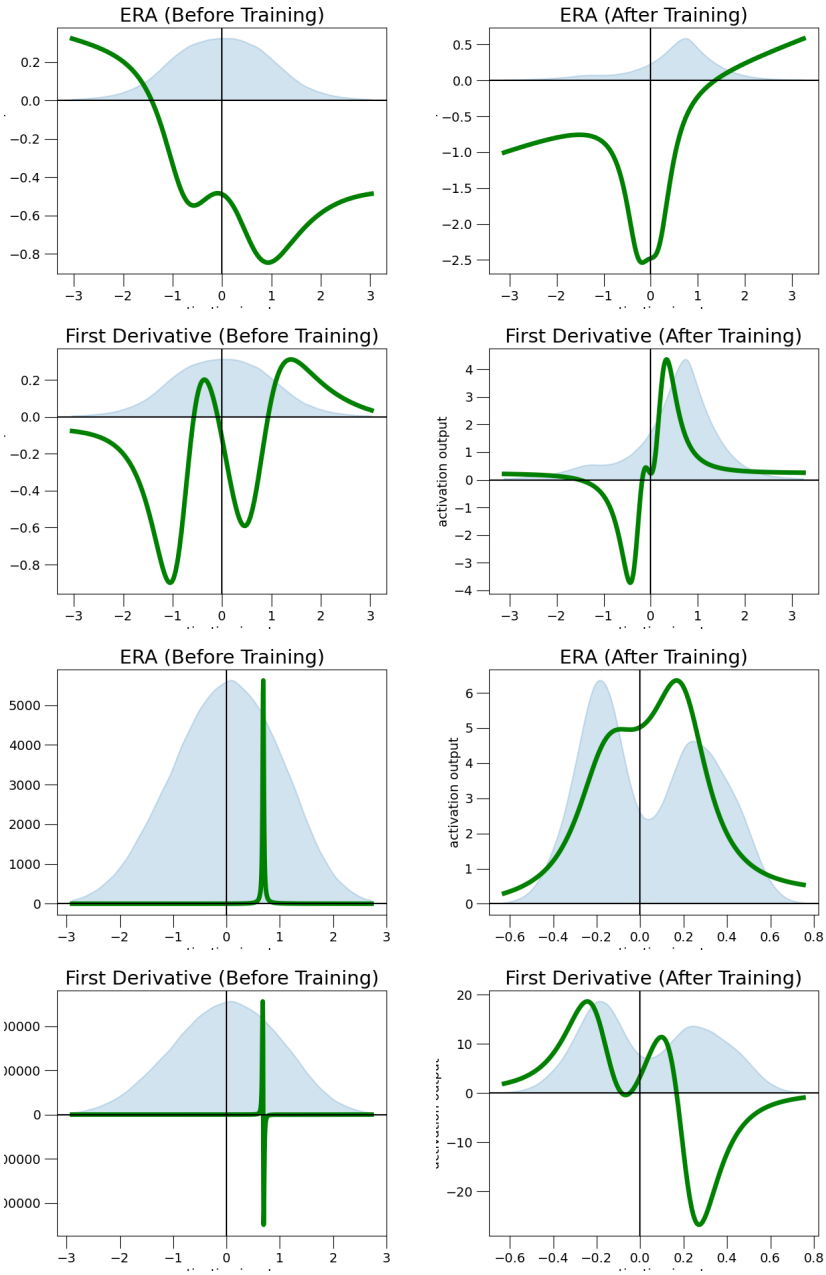

Fig. 2: Plots of the second (row 1 & 2) and third (row 3 & 4) *ERA* of the same CNN as in Figure 2 in the main paper. The *ERA* is randomly initialised and has degree 5/4. The shaded curve in the background shows the density distribution of the input to the activation (not true to scale on y-axis).

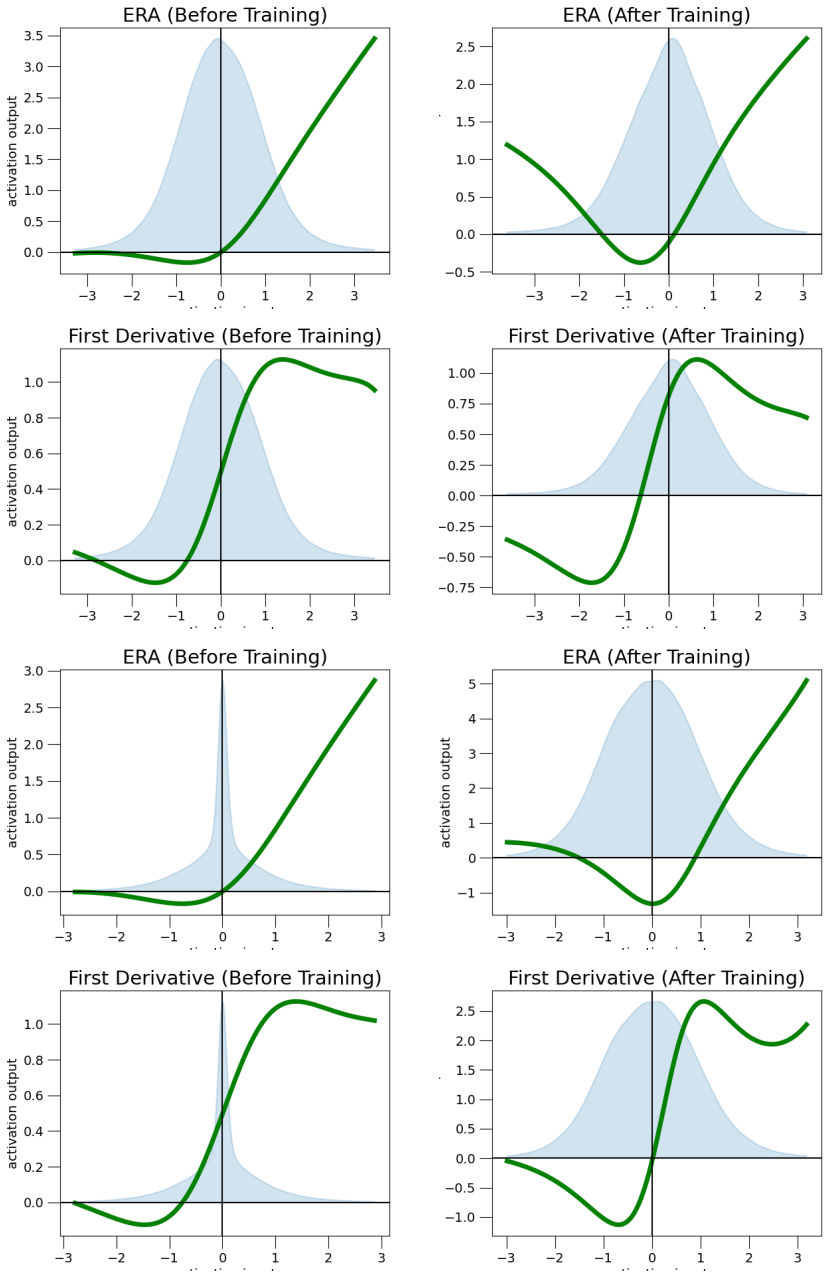

Fig. 3: Plots of the first (row 1 & 2) and second (row 3 & 4) *ERA* of a CNN re-trained with the same configuration in Figure 1 in the main paper. Initialisation: Swish, degree 5/4. The shaded curve in the background shows the density distribution of the input to the activation (not true to scale on y-axis).

Fig. 4: Plots of the third *ERA* of a CNN re-trained with the same configuration in Figure 1 in the main paper. The *ERA* is initialised as a Swish activation and has degree 5/4. The shaded curve in the background shows the probability density distribution of the input to the activation (not true to scale on the y-axis).

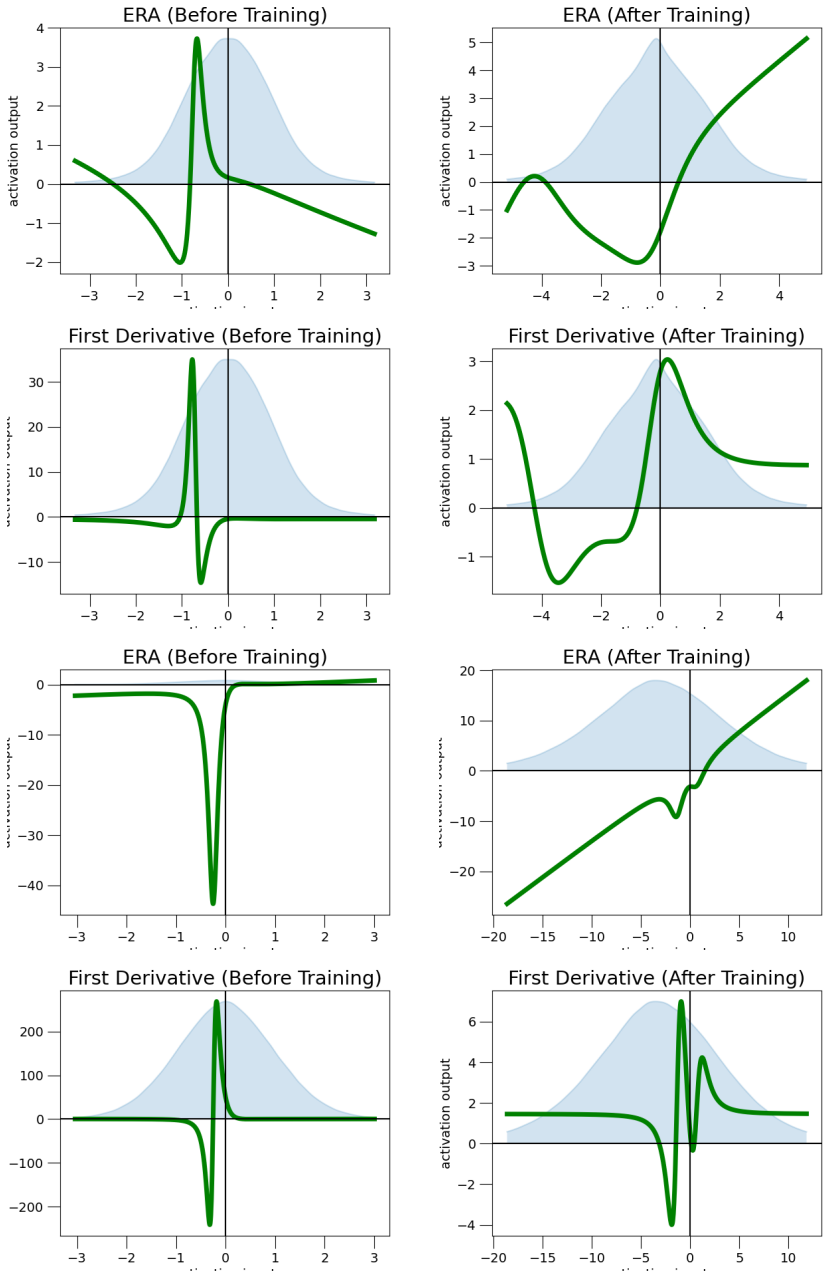

Fig. 5: Plots of the first (row 1 & 2) and second (row 3 & 4) *ERA* of a CNN retrained with the same configuration in Figure 2 in the main paper. Initialisation: random, degree: 5/4. The shaded curve in the background shows the density distribution of the input to the activation (not true to scale on y-axis).

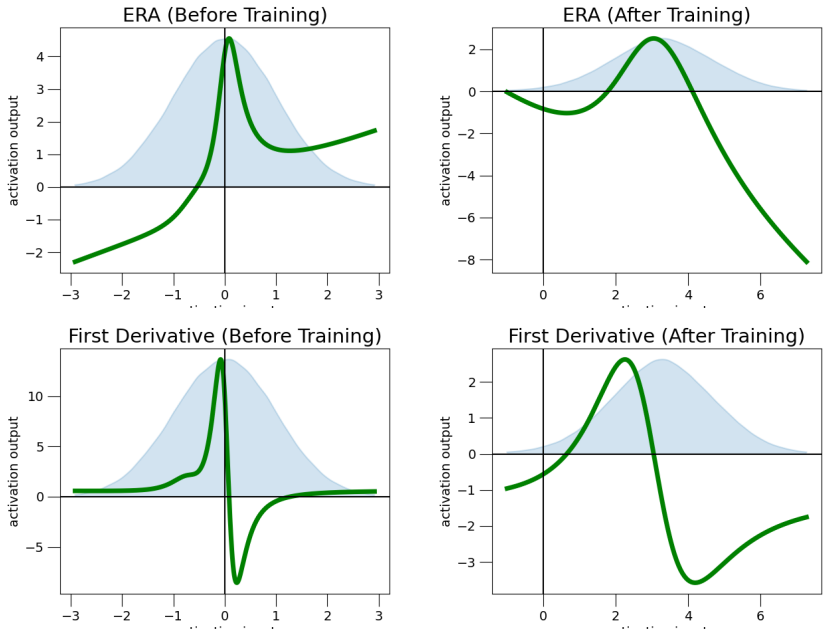

Fig. 6: Plots of the third *ERA* of a CNN re-trained with the same configuration in Figure 2 in the main paper. The *ERA* is randomly initialised and has degree 5/4. The shaded curve in the background shows the probability density distribution of the input to the activation (not true to scale on the y-axis).

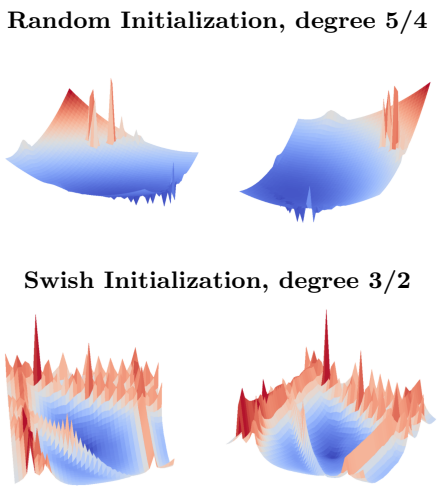

Fig. 7: Plots of the loss surfaces of neural networks trained on CIFAR10, using *ERA*. **Top**: CNN, using randomly initialized *ERA* of degree 5/4. Training did not converge. **Bottom**: FCN, using swish-initialized *ERA* of degree 3/2. Training converged. In each row, the two images show the same loss surface from different viewing angles.

benefit when using Swish initialisation.

Effect of Normalization Tables 2 and 3 show the effects of the learnable axes in the normalization layers. We conclude that for training is most stable and yields best results when layer normalization is used in fully-connected networks and instance normalization is used in convolutional neural networks. For CNNs, we have also experimented with different axes along which the standardization happens, but this had a strong adverse effect on training stability.

A.2 Results on MNIST

Tables 4 and 5 show our results on the MNIST dataset. Since image classification on MNIST is a very simple task, we reach 100% training accuracy with any setting that we tried. As a result, the corresponding test accuracies are all very similar. This underlines our observation that rational activation functions are most beneficial when the classification problem is hard and when the capacity of the network is limited. Interestingly, we observe that despite their potential to increase the network capacity, rational activation functions are not more prone to overfitting on simple data than standard activation functions. On the contrary, the best *ERA* being initialised as a Swish activation, obtains a test accuracy on par with the best non-rational activation function.

Network Type	**Initialisation**	**deg Q**	**No grad clipping**	**Grad clipping**
FCN	Random	2	40.5%	**57.0%**
FCN	Random	4	23.6%	**57.4%**
FCN	Swish	2	58.6%	**59.9%**
FCN	Swish	4	58.8%	**60.1%**
CNN	Random	2	68.6%	**81.4%**
CNN	Random	4	59.2%	**80.7%**
CNN	Swish	2	**84.0%**	83.1%
CNN	Swish	4	**84.3%**	83.0%

Table 1: Examining the effect of gradient clipping on the performance of the network after training. All networks were trained on CIFAR10. Gradient clipping greatly stabilises the training of randomly initialised *ERA*s, but it does not seem to offer a benefit when using Swish initialisation.

Initialisation	**deg Q**	**Axis -**	**Axis 3**
Random	2	48.7%	**48.8%**
Random	4	33.4%	**39.0%**
Swish	2	58.9%	**59.6%**
Swish	4	48.4%	**60.8%**

Table 2: Effect on the learnable axis on the test accuracy of fully-connected networks (FCNs) trained on CIFAR10. Axis - means that there is a single β and a single γ parameter learned for rescaling and shifting all the nodes of a given layer. Axis 3 means that there is one β and one γ parameter learned for each node of a given layer (this corresponds to standard layer normalization in TensorFlow). The latter shows consistent benefits across different initialisation types and rational function degrees.

Initialisation	**deg Q**	**Axis -**	**Axis 1**
Random	2	69.6%	**80.4%**
Random	4	51.0%	**54.7%**
Swish	2	83.5%	**83.6%**
Swish	4	64.2%	**65.9%**

Table 3: Effect on the learnable axis on the test accuracy of convolutional neural networks (CNNs) trained on CIFAR10. Axis - means that there is a single β and a single γ parameter learned for rescaling and shifting all the nodes of a given layer. Axis 3 means that there is one β and one γ parameter learned for each channel of a given layer (this corresponds to standard instance normalization in TensorFlow). The latter shows consistent benefits across different initialisation types and rational function degrees.

Activation	**Initialisation**	**deg Q**	**Training Accuracy**	**Test Accuracy**
GELU	-	-	100.0%	99.3%
Leaky ReLU	-	-	100.0%	99.2%
ReLU	-	-	100.0%	99.3%
Swish	-	-	100.0%	99.2%
Rational	GELU	2	100.0%	99.3%
Rational	GELU	4	100.0%	99.2%
Rational	Leaky ReLU	2	100.0%	99.2%
Rational	Leaky ReLU	4	100.0%	99.2%
Rational	Random	2	100.0%	99.1%
Rational	Random	4	100.0%	99.1%
Rational	ReLU	2	100.0%	99.3%
Rational	ReLU	4	100.0%	99.2%
Rational	Swish	2	100.0%	99.3%
Rational	Swish	4	100.0%	99.3%

Table 4: The results of training a convolutional neural network (CNN) with different activation functions on MNIST. Due to the simplicity of the dataset, all the activation functions perform similarly well. This is in line with our working hypothesis that rational activations have the greatest benefit when the network is lacking the capacity to solve a problem.

Activation	**Initialisation**	**deg Q**	**Training Accuracy**	**Test Accuracy**
GELU	-	-	100.0%	98.5%
Leaky ReLU	-	-	100.0%	98.3%
ReLU	-	-	100.0%	98.4%
Swish	-	-	100.0%	98.5%
Rational	GELU	2	100.0%	98.7%
Rational	GELU	4	100.0%	98.5%
Rational	Leaky ReLU	2	100.0%	98.5%
Rational	Leaky ReLU	4	100.0%	98.3%
Rational	Random	2	100.0%	98.2%
Rational	Random	4	100.0%	98.2%
Rational	ReLU	2	100.0%	98.5%
Rational	ReLU	4	100.0%	98.3%
Rational	Swish	2	100.0%	98.5%
Rational	Swish	4	100.0%	98.5%

Table 5: The results of training a fully-connected neural network (FCN) with different activation functions on MNIST. Due to the simplicity of the dataset, all the activation functions perform similarly well. This is in line with our working hypothesis that rational activations have the greatest benefit when the network is lacking the capacity to solve a problem.

A.3 Results on ImageNet

Our results on ImageNet are shown in Table 6 and Figure 8. In line with the experiments on CIFAR10 presented in the main paper, *ERA*s with Swish initialisation outperform all other activations.

Activation Function	**Training Accuracy**	**Test Accuracy**
ReLU	47.4%	38.4%
Swish	48.9%	40.1%
ERA(Random init)	47.6%	39.5%
ERA(Swish init)	50.5%	**40.7%**

Table 6: Results of training a lightweight ResNet-18 on the ILSVRC 2012 ("ImageNet") dataset, showing the same experiment as in Figure 8. In line with our results on CIFAR10, the best performing activation is a Swish-initialised *ERA*. Although the randomly initialised variant has a lower test accuracy than the standard Swish activation, it outperforms the ReLU activation.

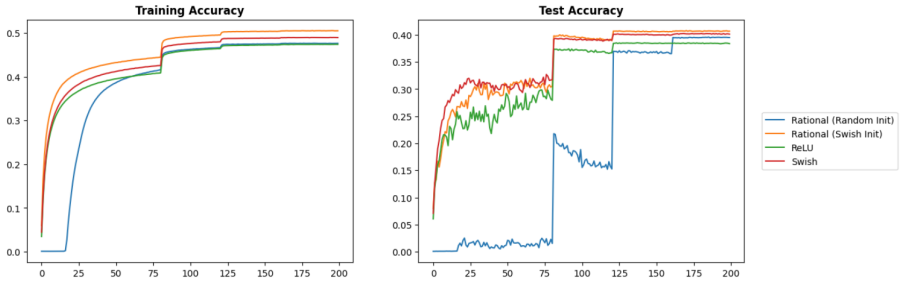

Fig. 8: Training and "test" (validation) accuracy of training a lightweight ResNet-16 on the ILSVRC 2012 ("ImageNet") dataset, showing the same experiment as in Table 6. The jumps in test accuracy for the randomly initialised *ERA* coincide with the epochs when the learning rate is reduced by a factor of 10.

B 3D Human Pose and Shape Reconstruction

Implementation details. We train our networks with a learning rate of $1e-4$, with exponential decay of 0.99 every 4000 steps. We use an ADAM optimizer. For the Transformer architecture, we use 4 encoder layers with 8 heads. We set the hidden dimension of the FFN to 1024. For the MLP-Mixer we use 4 mixer layers, with the tokens hidden dimension set to 1024 and 2048 for the channels. The size of the input images is 480×480. We train the MarkerPoser architecture, that translates from 3d marker positions to the GHUM[12] model space, as described in [13]. The network has around 920k parameters.

Network capacity of state-of-the-art methods. We show in Table 7 the capacities for competing methods on the Human3.6M dataset. Our proposed architectures have significantly less parameters, by one or even two orders of magnitude.

Method	**#Parameters**
HMR [5]	27.0M
GraphCMR [7]	42.7M
Pose2Mesh [2]	76.0M
I2L-MeshNet [11]	142.0M
SPIN [6]	27.0M
METRO [10]	243.0M
THUNDR [13]	25.0M

Table 7: Network capacity (i.e. number of trainable parameters) of state-of-the-art methods. Note that our proposed architectures are one or two orders of magnitude lower in capacity.

B.1 Additional ablations

Benefit of using *ERA* as a function of network capacity. We perform a more in detail experiment, where we sweep over the capacity of a Transformer (i.e. the hidden dimension) and show that the lower the capacity, the incresed benefit of *ERA* relative to other standard activation functions. In Figure 9 we present the MPJPE and MPJPE-PA metrics in a weakly supervised training regime, reported on the Human3.6M test dataset.

Shared encoder/mixer layer weights. In the original work of [13], the weights of the encoder layer for the Transformer were shared. We ablate this choice for both Transformer and MLP-Mixer architectures. In Table 8 we show the new number of parameters for our architectures, significantly lower than those we experimented with in the main paper. In Table 9 we show the results in a mixed training regime on the Human3.6M dataset. Our proposed activation

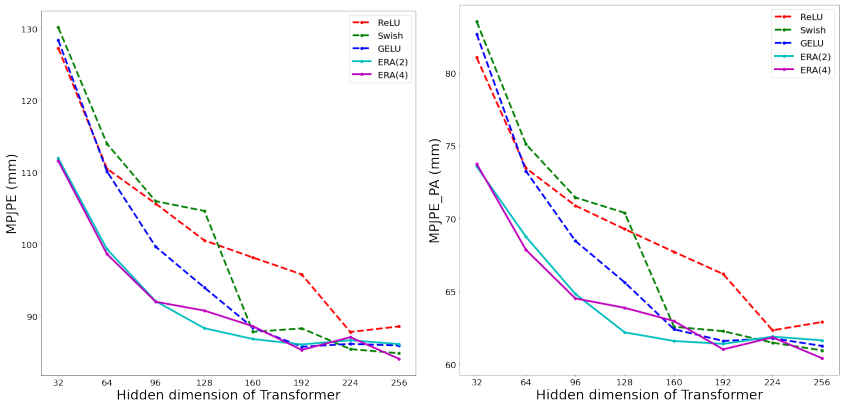

Fig. 9: Reconstruction errors as a function of the number of hidden dimension of the Transformer, for different activation functions. For our proposed activations, the number between brackets represents the degree of the denominator. Results are averaged over 3 trials, reported on the Human3.6M test dataset, protocol P2.

ERA boosts the performance of the network, especially in low-capacity settings. Results are overall improved over those presented in the main paper, showing that the choice of sharing weights is beneficial.

Backbone	**Head**	**Hidden dim**	**#Parameters**
MobileNet	MLP-Mixer	64	4.1M
MobileNet	MLP-Mixer	256	5.1M
MobileNet	Transformer	64	3.5M
MobileNet	Transformer	256	4.3M
ResNet50	Transformer	256	*25.0M* [13]

Table 8: Number of parameters for our lightweight versions of the architecture introduced in THUNDR [13], for the task of 3d pose and shape reconstruction. The weights for the encoder/mixer layers are **shared**.

Activation	**deg Q**	**Hidden dim**	**MPJPE-PA**	**MPJPE**	**MPVPE**
GELU	-	64	54.7 ±1.1	75.0±2.5	79.4±2.5
ReLU	-	64	51.4 ±0.9	71.0±0.6	75.6±0.2
Swish	-	64	52.5 ±0.1	70.1±0.4	78.2±0.8
ERA	2	64	**50.9** ±0.8	70.3 ±2.2	75.1±2.4
ERA	4	64	**50.9** ±0.6	**69.7**±2.0	**74.1**±2.2
GELU	-	256	48.2±0.5	67.3±1.7	71.8±1.5
ReLU	-	256	48.8±1.4	66.8±2.4	70.9±2.7
Swish	-	256	N/A	N/A	N/A
ERA	2	256	48.2±0.9	66.2±1.4	71.0±1.3
ERA	4	256	**47.7**±0.3	**66.1**±0.6	**70.6**±0.8

MLP-Mixer

Activation	**deg Q**	**Hidden dim**	**MPJPE-PA**	**MPJPE**	**MPVPE**
GELU	-	64	50.8±2.4	70.5±2.7	75.4±2.5
ReLU	-	64	50.0±1.6	70.1±3.2	74.7±3.0
Swish	-	64	N/A	N/A	N/A
ERA	2	64	47.5±1.0	66.9±1.4	71.3±1.0
ERA	4	64	**47.3**±1.1	**65.8**±0.4	**70.3**±0.1
GELU	-	256	43.3±0.8	61.1±1.1	64.9±0.6
ReLU	-	256	43.9±0.7	62.0±0.5	66.3±0.1
Swish	-	256	43.4±0.9	60.8±0.4	65.4±0.3
ERA	2	256	43.3±0.2	61.5±0.8	65.7±0.7
ERA	4	256	**42.5**±0.7	**60.4**±0.8	**64.6**±1.2

Transformer

Table 9: Ablation for networks with **MLP-Mixer (Top)** and **Transformer (Bottom)** architectures with a **shared** mixer or encoder layer, respectively. These are pre-trained in a weakly supervised regime and fine-tuned in a **fully-supervised** regime. Results are in mm, averaged over 3 trials, reported on the Human3.6M test dataset, protocol P2. We mark the **best** and second best results. *For some experiments with the Swish activation, the training diverged and so results are marked with "N/A".*

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/, software available from tensorflow.org
2. Choi, H., Moon, G., Lee, K.M.: Pose2mesh: Graph convolutional network for 3d human pose and mesh recovery from a 2d human pose. In: ECCV. pp. 769–787. Springer (2020)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database (2009). https://doi.org/10.1109/CVPR.2009.5206848
4. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR **abs/1704.04861** (2017), http://arxiv.org/abs/1704.04861
5. Kanazawa, A., Black, M.J., Jacobs, D.W., Malik, J.: End-to-end recovery of human shape and pose. In: CVPR (2018)
6. Kolotouros, N., Pavlakos, G., Black, M.J., Daniilidis, K.: Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2252–2261 (2019)
7. Kolotouros, N., Pavlakos, G., Daniilidis, K.: Convolutional mesh regression for single-image human shape reconstruction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4501–4510 (2019)
8. Krizhevsky, A.: Learning multiple layers of features from tiny images. Master's thesis, University of Toronto (2009)
9. LeCun, Y., Cortes, C., Burges, C.: Mnist handwritten digit database. ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist (2010)
10. Lin, K., Wang, L., Liu, Z.: End-to-end human pose and mesh reconstruction with transformers. In: CVPR. pp. 1954–1963 (2021)
11. Moon, G., Lee, K.M.: I2l-meshnet: Image-to-lixel prediction network for accurate 3d human pose and mesh estimation from a single rgb image. In: European Conference on Computer Vision (ECCV) (2020)
12. Xu, H., Bazavan, E.G., Zanfir, A., Freeman, W.T., Sukthankar, R., Sminchisescu, C.: Ghum & ghuml: Generative 3d human shape and articulated pose models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6184–6193 (2020)
13. Zanfir, M., Zanfir, A., Bazavan, E.G., Freeman, W.T., Sukthankar, R., Sminchisescu, C.: Thundr: Transformer-based 3d human reconstruction with markers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 12971–12980 (October 2021)