Singh, S. P. Design and Fabrication of Microstrip Patch Antenna at 2.4 Ghz for WLAN Application using HFSS. *IOSR Journal of Electronics and Communication Engineering*, pp. 01–06, January 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

von Oswald, J., Henning, C., Sacramento, J., and Grewe, B. F. Continual learning with hypernetworks. In *International Conference on Learning Representations*, 2020.

Wheeler, H. Small antennas. *IEEE Transactions on Antennas and Propagation*, 23(4):462–469, 1975.

Zhou Wang, E. P. S. and Bovik, A. C. Multi-scale structural similarity for image quality assessment. *IEEE 37th Asilomar*, 2003.

A. Experimental Results

a sample array design was sent out for manufacturing Fig. 6. (i) The simulated antenna and the fabricated one agree with a 1dB difference in terms of directivity as measured in an anechoic chamber. (ii) The measured antenna array efficiency,$\eta = 99.6\%$. (iii) Connected to a Galaxy phone that has also debug telemetries from a cellular modem for testing, the logs extracted from the phone show that with the manufactured antenna array the SNR of receiving the cell signals improves,in comparison to the phones original antenna, by 10dB.

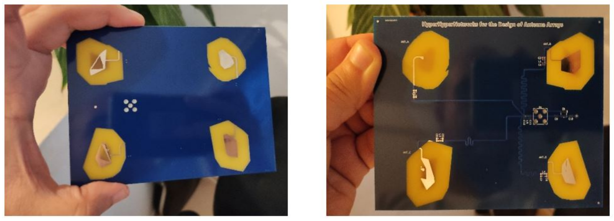


Figure 6. Manufactured antenna array. Front and back views.

B. Signal Model

In order to select the number of antennas for the antenna array use case, we analyze a specific yet common specification of the mobile channel and its physical layer. For the received signal model, we assuming a QPSK transmitted narrowband symbol and Short Observation Interval Approximation (SOIA approximation), propagate through Rician channel (Roberts & Abeysinghe, 1995). The justification for QPSK signal can be seen from the vast majority of protocols using it[2].

[2]Tektronix Overview of 802.11 Physical Layer, https://download.tek.com/document/37W-29447-2_LR.pdf

Let s be a QPSK symbol, i.e.

$$s \in [1+j, 1-j, -1+j, -1-j]/\sqrt{2} \tag{17}$$

Let subscript i denote the number of receiving antenna in the array. The physical channel h_i can be modeled as a complex phasor for the i-th antenna physical channel (Roberts & Abeysinghe, 1995).

$$h_i \sim rice(K, \Omega) \tag{18}$$

Where K is the ratio between the direct path and the reflections, Ω is the total received power. The noise is modeled as complex normal additive noise,

$$n \sim CN(0, \sigma^2) \tag{19}$$

The received signal at the i-th antenna reads,

$$s_i = h_i \cdots + n \tag{20}$$

The MIMO scheme chosen for the evaluation task is the Multi Rate Combiner (MRC) (Poor & Wornell, 1998), We define the Signal to Noise Ratio (SNR) as

$$SNR_i = \frac{|h_i|^2}{|\sigma^2|} \tag{21}$$

Thus the post-processing signal s_p reads

$$s_p = \frac{SNR_0 s_0 + ...SNR_i s_i}{SNR_0 + ...SNR_i} \tag{22}$$

Simulations Let $SNR_i \sim 4dB, K = 2.8, Delayspread = 58[ns]$, we ran a Monte-Carlo of Bit Error Rate (BER) as function of number of antennas in receiving.

From the simulation, it is observable that while increasing the number of antennas from 1 to 6 improves the Bit Error Rate (BER) by an order of magnitude, in order to achieve another order of magnitude improvement we are ought to increase the number of elements by three times. This implies that for the given (fairly common) scenario, 6 elements are a reasonable upper bound. We, therefore, set $N_{array} = 6$.

The Array Gain for the observation angle ϕ, θ reads

$$AG(\theta, \phi) = \sum_{ant} (G_{ant}(\theta, \phi) w_{ant} \exp(-jkr_{ant})) \tag{23}$$

$$k = 2\pi/\lambda \times [sin(\theta)cos(\phi), sin(\theta)sin(\phi), cos(\theta)] \tag{24}$$

Where $G_{ant}(\theta, \phi) \in \mathbb{R}$ is the real valued gain of an element in the array, $r_{ant} \in \mathbb{R}^3$ is the position of this element relative

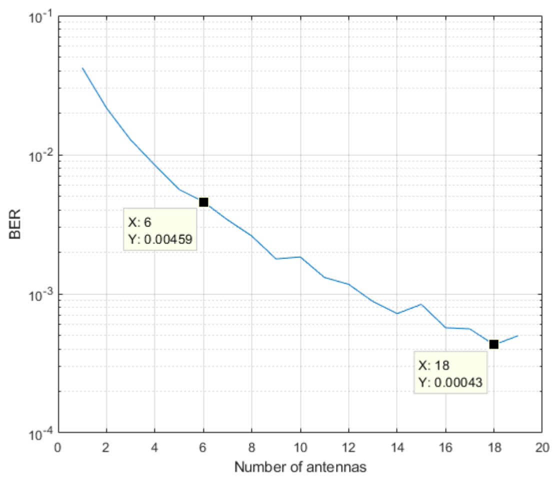


Figure 7. BER performance of MRC combiner with different number of antennas. With 10^5 Monte Carlo iterations.

to zero phase of the array. A natural selection for w_{ant} is the following beamforming coefficients

$$w_{ant} = exp(j\frac{2\pi}{\lambda}sin(\theta_d)(cos(\phi_d)r_x + sin(\phi_d)r_y)) \quad (25)$$

Where θ_d, ϕ_d is the beamforming angular direction. Choosing the centered steered array (zero direction) Eq. 23 reads

$$AG(\theta, \phi) = \sum_{ant}(G_{ant}(\theta, \phi)\exp(-jkr_{ant})) \quad (26)$$

C. Reproducibility

C.1. Dataset

Our synthetic dataset includes 3,000 examples of simulated multi-layer printed circuit board (PCB) antennas. We use (Liebig, 2010) engine with the MATLAB API to generate all of those examples. In order to span a wide range of designs we opt for random designs, We draw the following random parameters over uniform distribution : (i) number of polygons in each layer (ii) number of layers in the antenna (iii) number of cavities in the polygon. The feeding point is fixed during all simulations, as the dielectric constant. The dimensions of the antenna were also allowed to change in the scale of $[\frac{\lambda}{10}, \frac{\lambda}{4}]$.

For the array antennas a post-simulation process was done as stated in the article, where for each array training example the following parameters are chosen: (i) a Random number of elements over uniform probability $U(1, 6)$ (ii) Random position out of 6 predefined center phase locations (iii) Calculating the array gain, AG, according to Eq. 26.

C.2. Code

We use the PyTorch framework, all the models are given as separate modules: 'array_network_transformer','array_network_resnet', 'array_network_ablation', 'designer_resnet','desinger_transformer','simulator_network'. The training sequence 'array_training.py' is also given to give an example of running the models.

In our code we make a use of external repositories:

1. https://github.com/FrancescoSaverioZuppichini/ResNet,
2. https://github.com/VainF/pytorch-msssim (multi scale SSIM)
3. https://github.com/facebookresearch/detr (for the transformer's positional encoding)

The code for the block selection method is 'block_selector.py', which is a class that gets as input a PyTorch model with loss function, input, and constraints generators as input and outputs the entropy for each layer index.

C.3. Training

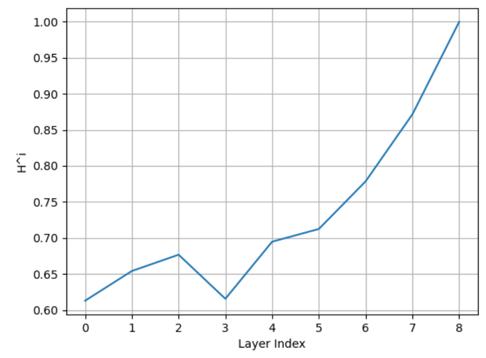
The training time of the different networks: Simulator network 10 hours, Single antenna Designer 3-7 hours (depending on the variant), Array designer 2-10 hours (depending on the variant). All the networks trained with Adam optimizer and learning rate of 10^{-4} with decay factor of 0.98 every two epochs. After training the simulator network h another 10^4 examples were generated within 5 hours. The dataset is then divided into train/test sets with a 90%-10% division.

C.4. Initialization details

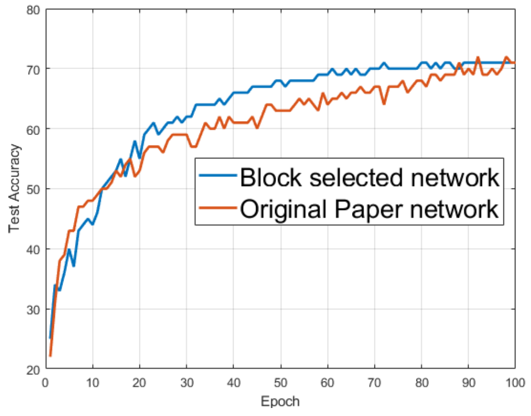
The initialization scheme we purposed assumes we have beforehand the variance of the input constraints. In order to estimate this variance, we calculate the mean value of the constraint variance over the train set. This mean valued variance is then used as a constant for the initialization method when constructing the network class.

C.5. Computing infrastructure

The generation of the 3,000 examples takes 2 weeks over a machine with I7-9870H CPU, 40GB RAM, and Nvidia RTX graphics card. In addition, another machine was in use with 16GB RAM and Nvidia P100 graphics card.



(a)



(b)

Figure 8. (a) Entropy loss vs layer index for the ”AllConv” network. (b) Test accuracy of our block selected method (blue) and original network (orange).

D. Block Selection Experiments on Hypernetworks

In order to further evaluate our heuristic method for selecting the important blocks, we evaluate it on one of the suggested networks of (Chang et al., 2020). Specifically, we test it on the ”All Conv” network, trained on the CIFAR-10 dataset.

Our block selection method first predicts the important layers, based on the entropy metric, then using a knapsack and chooses the most significant layers.

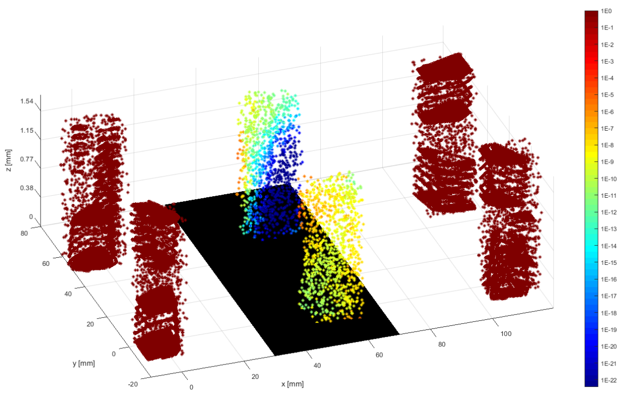
Fig. 8 shows (a) the entropy metric (normalized to the maximal value) for ”AllConv” network, (b) Test accuracy for both networks. Using only 2.8% of the weights (first and last layer) as the output of the hypernetwork, the suggested network is able to achieve the same results over the test set. The total size of the network is reduced by a factor of 3.4.

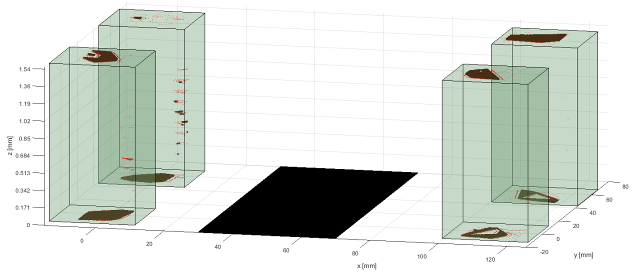
E. Additional Figures

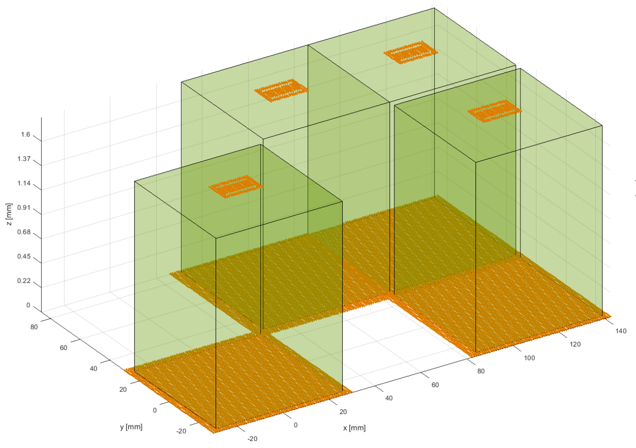
Fig. 9 is an enlarged version of Fig. 5 from the main paper. This figure also includes the results of the ablation experiments (panels e,f), which were omitted from the main paper for brevity.

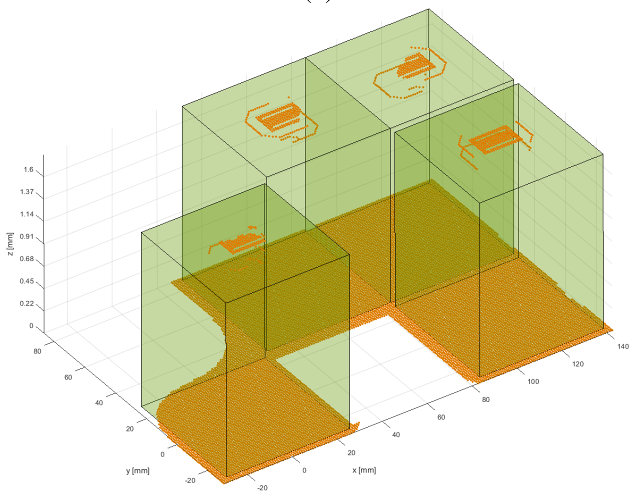
Fig. 10 shows the effect of the different initialization schemes for the Transformer architecture (same as papers Fig. 4(a) but for a Transformer instead of a ResNet).

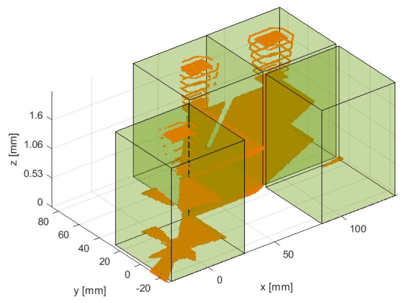
Fig. 11 shows the computed entropy of different layers un-normalized, whereas in the main manuscript Fig. 4(b) depicted the same score normalized by the maximum over the different layers.


(a)


(b)


(c)


(d)


(e)

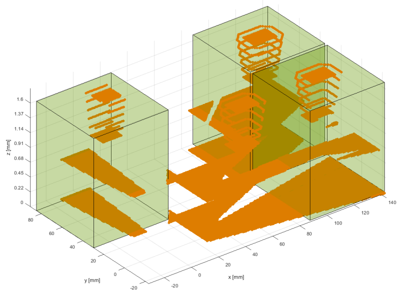
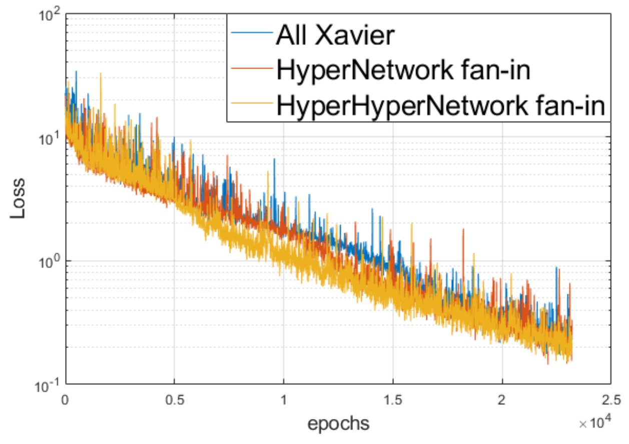

(f)

Figure 9. (a) The probability of points belong to a valid antenna in a synthetic test instance. The constraint plane is marked as black. (b) Same sample, the regions correctly classified as antenna are marked in brown, misclassified is marked in red. (c) The ground truth of a slotted antenna array. (d) Our network design. (e) The design of ablation (i) that does not use a hyperhypernetwork. (f) The design of ablation (ii) that does not use a hypernetwork.



(a)

Figure 10. Loss per epochs for the different initialization scheme of q (Transformer f).

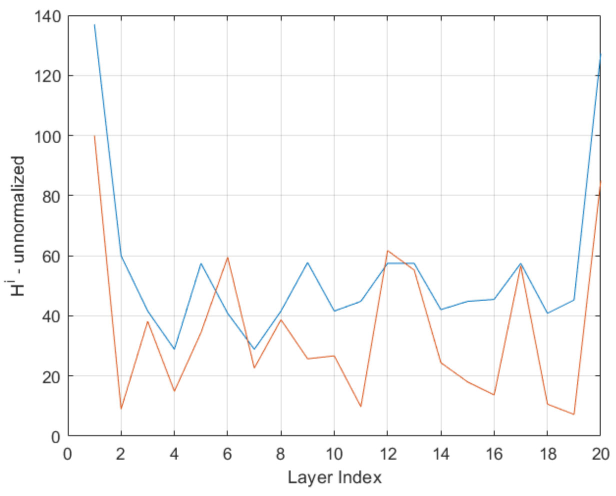


Figure 11. Un-normalized (relative to maximal value) Entropy of both ResNet and Transformer networks.