Appendix for
Loss Surface Simplexes for Mode Connecting Volumes and Fast Ensembling

Outline

The Appendix is outlined as follows:

In Appendix A.1, we give a more detailed description of our methods, focusing first on computing the simplex volume and sampling from the simplexes, then describe vertex initialization and regularization, giving training details, and finally describing the training procedure for multi-dimensional mode connectors.

In Appendix A.2, we describe several more results on volume and ensembling, particularly on the number of samples required for good performance with SPRO and ESPRO.

Finally, in Appendix C, we plot the results of a larger suite of corruptions on CIFAR-10 for ESPRO, deep ensembles, and MultiSWAG.

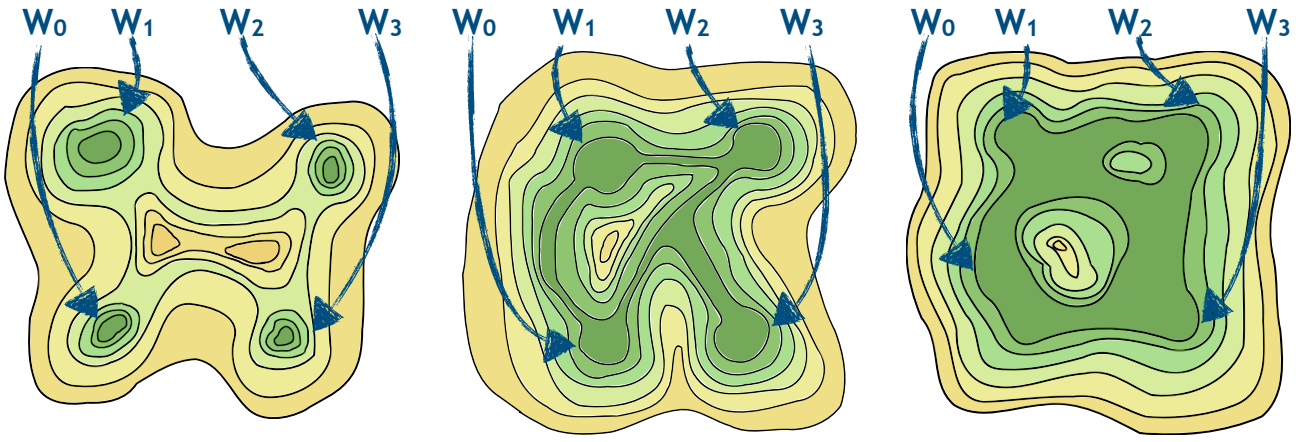


Figure A.1. A simplified version of the progressive understanding of the loss landscape of neural networks. **Left:** The traditional view in which low loss modes are disconnected in parameter space. **Center:** The updated understanding provided by works such as Draxler et al. (2018), Fort & Jastrzebski (2019), and Garipov et al. (2018), in which modes are connected along thin paths or tunnels. **Right:** The view we present in this work: independently trained models converge to points on the same *volume* of low loss.

A. Extended Methodology

First, we present a two dimensional version of the schematic in Figure 1 in A.1, which explains the same progressive illustration, but in two dimensions.

A.1. Simplex Volume and Sampling

We employ simplexes in the loss surface for two reasons primarily:

- sampling uniformly from within a simplex is straightforward, meaning we can estimate the expected loss within any found simplexes easily,
- computing the Volume of a simplex is efficient, allowing for regularization encouraging high-Volume simplexes.

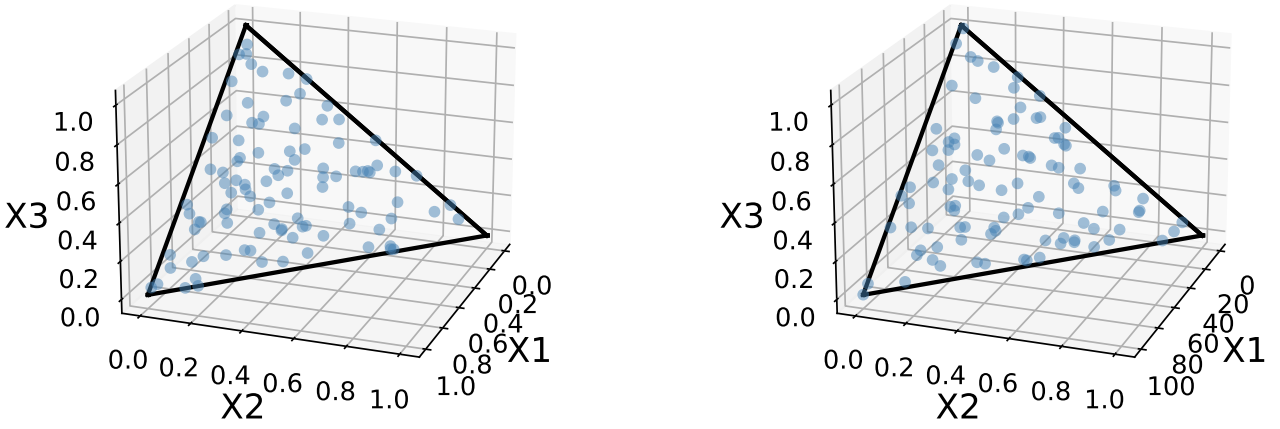


Figure A.2. **Left**: 100 samples drawn uniformly from within the unit simplex. **Right:** 100 samples drawn from a non-unit simplex (note the scale of the $X1$ axis). The distribution of points in both simplexes is visually indistinguishable — evidence that the method for sampling from a unit simplex is sufficient to draw samples from arbitrary simplexes.

Sampling from Simplexes: Sampling from the standard simplex is just a specific case of sampling from a Dirichlet distribution with concentration parameters all equal to 1. The standard n-simplex is a simplex is a simplex formed by the vectors $\mathbf{v}_0, \ldots, \mathbf{v}_n$ such that the $\mathbf{v}_i$'s are the standard unit vectors. Therefore, to draw samples from a standard n-simplex in a d dimensional space with vertices $\mathbf{v}_0, \ldots, \mathbf{v}_n$, we follow the same procedure to sample from a Dirichlet distribution.

To sample vector $\mathbf{x} = [x_0, \ldots, x_d]^T$ we first draw $y_0, \ldots, y_n \overset{\text{i.i.d.}}{\sim} \text{Exp}(1)$, then set $\tilde{y}_i = \frac{y_i}{\sum_{j=1}^{d} y_j}$. Finally, $\mathbf{x} = \sum_{i=1}^{n} \tilde{y}_i \mathbf{v}_i$.

While this method is sufficient for simulating vectors uniformly at random from the *standard* simplex, there is no guarantee that such a sampling method produces uniform samples from an arbitrary simplex, and thus samples of the loss over the simplex that we use in Equation 1 may not be an unbiased estimate of the expected loss over the simplex. Practically, we do not find this to be an issue, and are still able to recover low loss simplexes with this approach.

Furthermore, Figure A.2 shows that the distribution of samples in a unit simplex is visually similar to the samples from an elongated simplex where we multiply one of the basis vectors by a factor of 100. This figure serves to show that although there may be some bias in our estimate of the loss over the simplex in Equation 1, it should not be (and is not in practice) limiting to our optimization routine. Note too, this may appear like a simplistic case, but typically the simplexes found by SPRO contain only a small number of vertices, so a 2-simplex whose edge lengths vary by a factor of nearly 100 is a reasonable comparison to a scenario we may find in practice.

Computing Simplex Volume: Simplex Volumes can be easily computed using Cayley-Menger determinants (Colins). If we have an n-simplex defined by the parameter vectors $w_0, \ldots, w_n$ the Cayley-Menger determinant is defined as

$$CM(w_0, \ldots, w_n) = \begin{vmatrix} 0 & d_{01}^2 & \cdots & d_{0n}^2 & 1 \\ d_{01}^2 & 0 & \cdots & d_{0n}^2 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ d_{n0}^2 & d_{n1}^2 & \cdots & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{vmatrix}. \tag{A.1}$$

The Volume of the simplex $S_{(w_0,\ldots,w_n)}$ is then given as

$$\text{V}(S_{w_0,\ldots,w_n})^2 = \frac{(-1)^{n+1}}{(n!)^2 2^n} CM(w_0, \ldots, w_n). \tag{A.2}$$

While in general we may be adverse to computing determinants or factorial terms the simplexes we work with in this paper

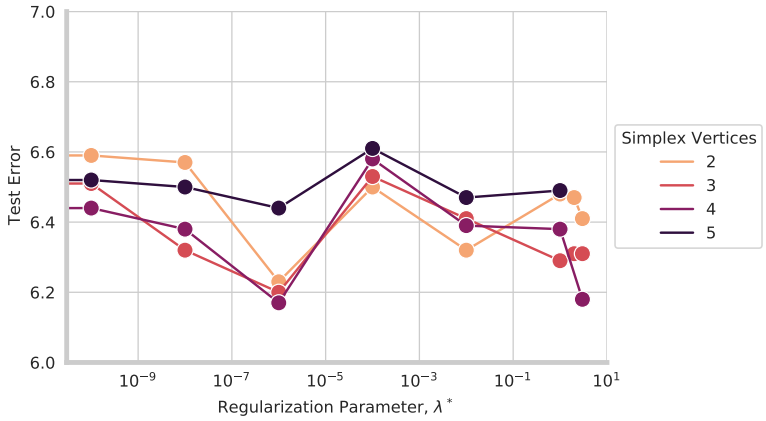


Figure A.3. CIFAR-10 test accuracy as a function of regularization parameter λ^* and colored by the number of vertices. Accuracy is essentially unchanged for the various regularization parameters.

are generally low order (all are under 10 vertices total) meaning that computing the Cayley-Menger determinants is generally a quite fast and stable computation.

A.2. Initialization and Regularization

Vertex Initialization: We initialize the j^{th} parameter vector corresponding to a vertex in the simplex as the mean of the previously found vertices, $w_j = \frac{1}{j}\sum_{i=0}^{j-1} w_i$ and train using the regularized loss in Eq. 1.

Regularization Parameter: As the order of the simplex increases, the Volume of the simplex increases exponentially. Thus, we define a distinct regularization parameter, λ_j, in training each θ_j to provide consistent regularization for all vertices. To choose the λ_k's we define a λ^* and compute

$$\lambda_k = \frac{\lambda^*}{\log \mathrm{V}(\mathcal{K})}, \tag{A.3}$$

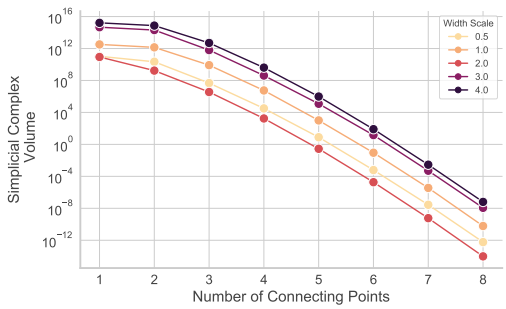
where $\mathcal{K}$ is randomly initialized simplicial complex of the same structure that the simplicial complex will have while training θ_j. Eq. A.3 normalizes the λ_k's such that they are similar when accounting for the exponential growth in volume as the order of the simplex grows. In practice we need only small amounts of regularization, and choose $\lambda^* = 10^{-8}$. As we are spanning a space of near constant loss any level of regularization will encourage finding simplexes with non-trivial Volume.

Finally, when dealing with models that use batch normalization, we follow the procedure of Garipov et al. (2018) and compute several forwards passes on the training data for a given sample from the simplex to update the batch normalization statistics. For layer normalization, we do not need to use this procedure as layer norm updates at test time.

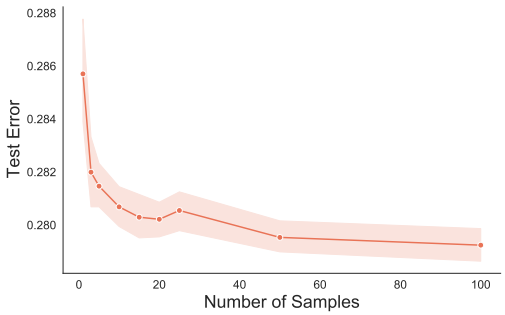
A.3. Training Details

We used VGG-16 like networks originally introduced in Simonyan & Zisserman (2015) from `https://github.com/pytorch/vision/blob/master/torchvision/models/vgg.py`. For training, we used standard normalization, random horizontal flips, and crops and a batch size of 128. We used SGD with momentum $= 0.9$, and a cosine annealing learning rate with a single cycle, a learning rate of 0.05, and weight decay $5e-4$ training for 300 epochs for the pre-trained VGG models. For SPRO, we used a learning rate of 0.01 and trained for 20 epochs for each connector.

In our experiments with transformers, we used the *ViT-B_16* image transformer model (Dosovitskiy et al., 2021) pre-trained on ImageNet from `https://github.com/jeonsworld/ViT-pytorch` and trained on CIFAR100 with upsampled image size of 224 with a batch size of 512 for 50000 steps (the default fine-tuning on CIFAR-100). Again, we used random flips and crops for data augmentation. To train these SPRO models, we used a learning rate of 0.001 and trained with SGD for 30 epochs for each connector, using 20 samples from the simplex at test time.



(a) Log volumes, MNIST.



(b) Test error vs. number of samples, CIFAR-100

Figure A.4. **(a)** Log volumes as a function of LeNet-5 layer width. Volumes are generally highest for wider models, and the volume of the simplicial complex tends to decrease as the dimension of the space increases. **(b)**Test error vs. number of samples, J, in the ensemble on CIFAR-100 using a VGG-16 network and a 3-simplex trained with SPRO. For any number of components in the SPRO ensemble greater than approximately 25 we achieve near constant test error.

A.4. Multi-Dimensional Mode Connectors

To train the multi-dimensional SWAG connectors, we connected two pre-trained networks following Garipov et al. (2018) using a piece-wise linear curve, trained for 75 epochs with an initial learning rate of 0.01, decaying the learning rate to $1e-4$ by epoch 40. At epoch 40, we reset the learning rate to be constant at $5e-3$. The final individual sample accuracy (not SWA) was 91.76%, which is similar to the final individual sample accuracies for standard training of VGG networks with SWAG. We used random crops and flips for data augmentation.

B. Extended Volume and Ensembling Results

B.1. Volumes on MNIST

In a similar construction to the dimensionality experiment in Figure 5, we next consider lower bounding the dimensionality of the connecting space that SPRO can find for LeNet-5s on MNIST (LeCun et al., 1998)[4], varying the width of the convolutional networks from a baseline of 1 (standard parameterization), either halving the width or consecutively widening the layers by a constant factor. We find in Figure A.4a that the volumes of the simplicial complex can vary by several powers of 10 for the as we increase the widths. However, all width networks generally follow the same patter of decaying volume as we increase the number of connecting points (e.g. increasing the dimensionality of the simplicial complex).

B.2. Test Error vs. Simplex Samples

SPRO gives us access to a whole space of model parameters to sample from rather than just a discrete number of models to use as in deep ensembles. Therefore a natural question to ask is how many models and forwards passes need to be sampled from the simplex to achieve the highest accuracy possible without incurring too high of a cost at test time.

Figure A.4b shows that for a VGG-16 network trained on CIFAR-100 we achieve near constant accuracy for any number of ensemble components greater than approximately 25. Therefore, for the ensembling experiments in Section 5.4 we use 25 samples from each simplex to generate the SPRO ensembles. In this work we are not focused on the issue of test time compute cost, and if that were a consideration for deployment of a SPRO model we could evaluate the trade-off in terms of test time compute vs accuracy, or employ more sophisticated methods such as ensemble distillation.

B.3. Loss Surfaces of Transformers

Next, we show the results of training a SPRO $3-$simplex with an image transformer on CIFAR-100 (Dosovitskiy et al., 2021) in Figure A.5. Due to computational requirements, the transformer was pre-trained on ImageNet before being fine-tuned on CIFAR-100 for $50,000$ minibatches. We then trained each vertex for an additional 10 epochs. Due to the inflexibility of the architecture, we observed training instability, which ultimately produced a small volume of the simplex

[4]Implementation from https://github.com/activatedgeek/LeNet-5/blob/master/lenet.py

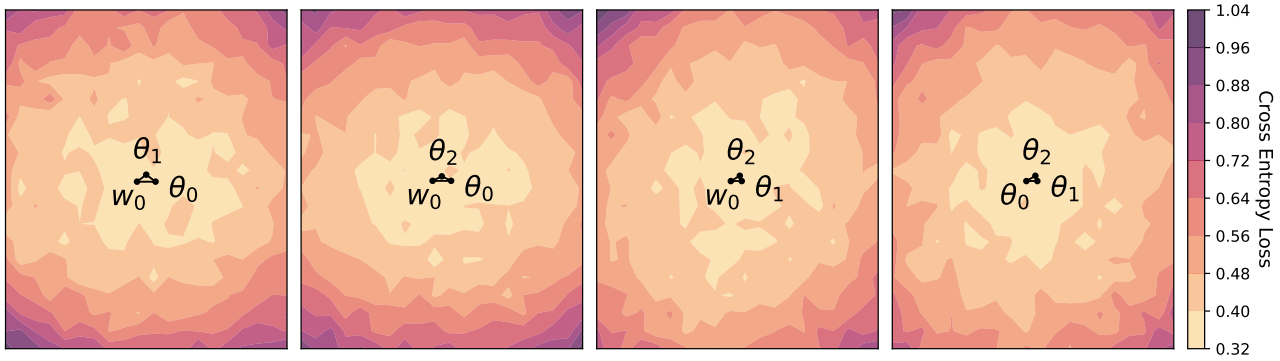


Figure A.5. Loss surface visualizations of the faces of a sample ESPRO 3-simplex for a Transformer architecture (Dosovitskiy et al., 2021) fine-tuned on CIFAR-100. Here, the volume is considerably smaller, but a low loss region is found.

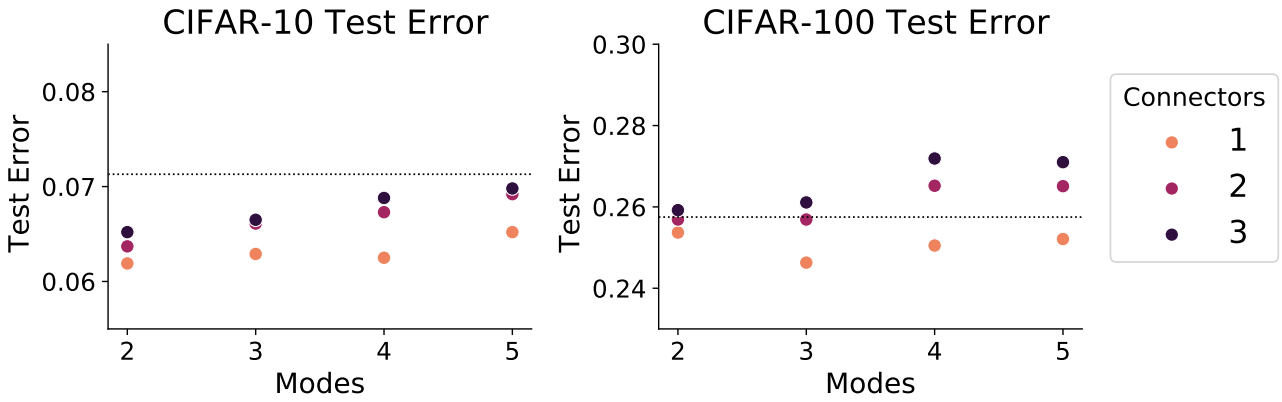


Figure A.6. Test error for mode connecting simplexes that connect various numbers of modes through various numbers of connecting points in the parameter space of VGG-16 networks trained on CIFAR-10 and CIFAR-100. The error rates of baseline models are shown as horizontal dotted lines. In general the highest performing models are those with the fewest modes and the fewest connecting points, but the performance gaps between configurations are small.

found (approximately 10^{-21}). Furthermore, the small volume of the simplex produced less diverse solutions, limiting the benefits of ensembling transformer models as shown in Figure A.8. However, these results demonstrate that a region of low loss can be found in subspaces of transformer models, and further work will be necessary to efficiently exploit these regions of low loss, much like has been done with CNNs and ResNets.

B.4. Ensembling Mode Connecting Simplexes

We can average predictions over these mode connecting volumes, generating predictions as ensembles, $\hat{y} = \frac{1}{H}\sum_{\phi_h \sim \mathcal{K}} f(x, \phi_h)$, where $\phi_h \sim \mathcal{K}$ indicates we are sampling models uniformly at random from the simplicial complex $\mathcal{K}(S_{(w_0,\theta_0,\dots)}, S_{(w_1,\theta_0,\dots)}, \dots)$. Test error for such ensembles for volumes in the parameter space of VGG-16 style networks on both CIFAR-10 and CIFAR-100 are given in Figure A.6. We see that while some improvements over baselines can be made, mode connecting simplexes do not lead to highly performant ensembles.

B.5. Ensembling Modes of SPRO

Figure A.7 presents the results of Figure 8 in the main text, but against the total training budget rather than the number of ensemble components. We see from the plot that on either dataset, for nearly any fixed training budget the most accurate model is an ESPRO model, even if that means using our budget to train ESPRO simplexes but fewer models overall.

Times correspond to training models sequentially on an NVIDIA Titan RTX with $24GB$ of memory.

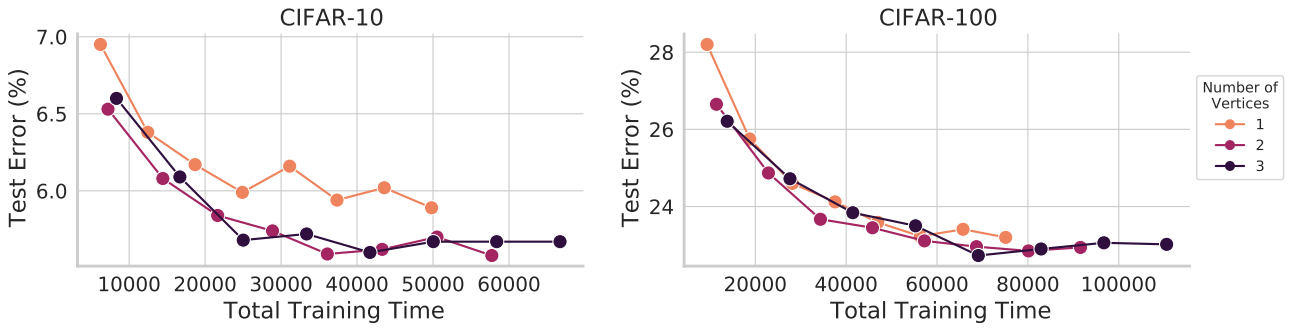


Figure A.7. Test error of ESPRO models on CIFAR-10 (left) and CIFAR-100 (right) as a function of total training time (training the original models and the ESPRO simplexes). The color of the curves indicate the number of the vertices in the simplex, and the points corresponding to increasing numbers of ensemble components moving left to right (ranging from 1 to 8). We see that on either dataset for nearly any fixed training budget, we are better off training fewer models overall and using ESPRO to construct simplexes to sample from.

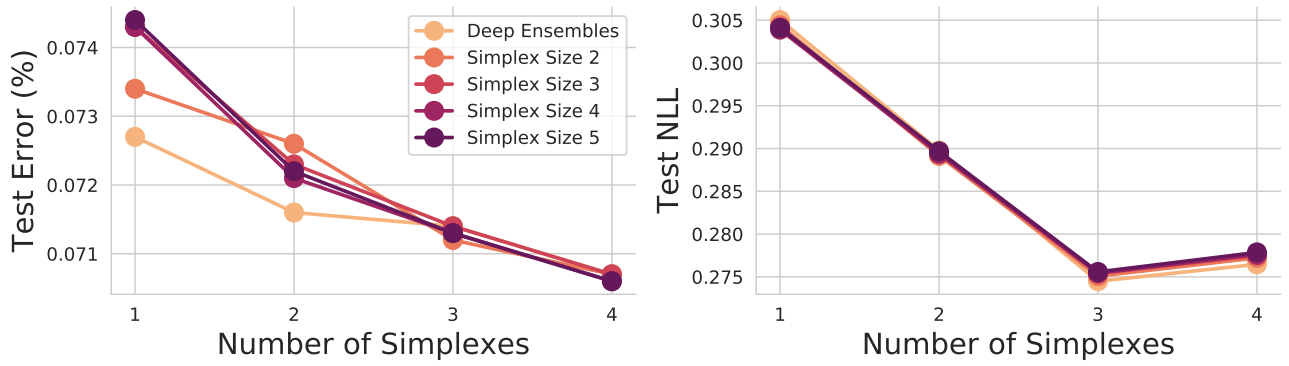


Figure A.8. Test Error and NLL for the number of components in SPRO ensembles using image transformers on CIFAR-100. ESPRO with four dimensional simplexes is slightly better in test accuracy and slightly worse in test NLL than deep ensembles.

Finally, Figure A.8 presents the results of ensembling with SPRO using state of the art transformers architectures on CIFAR-100 (Dosovitskiy et al., 2021). We find, counterintuitively that there is only a very small performance difference from ensembling with SPRO compared to the base architecture. We suspect that this is because it is currently quite difficult to train transformers without using significant amounts of unlabelled data.

C. Extended Uncertainty Results

C.1. Further NLL and Calibration Results

Finally, we include the results across 18 different corruptions for the ensemble components. In order, these are *jpeg*, fog, snow, brightness, pixelate, zoom blur, saturate, contrast, motion blur, defocus blur, speckle noise, gaussian blur, glass blur, shot noise, frost, spatter, impulse noise and, elastic transform.

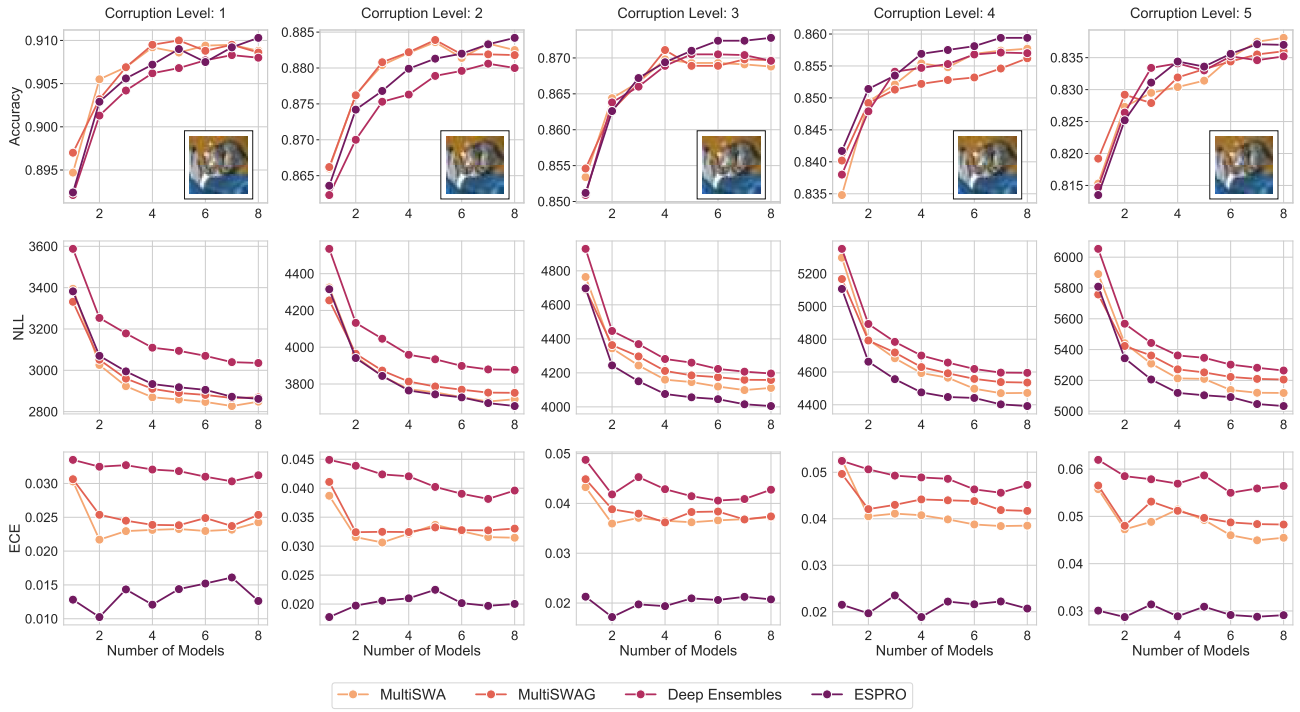


Figure A.9. Accuracy, NLL and ECE with increasing intensity of the *jpeg compression* corruption (from left to right).

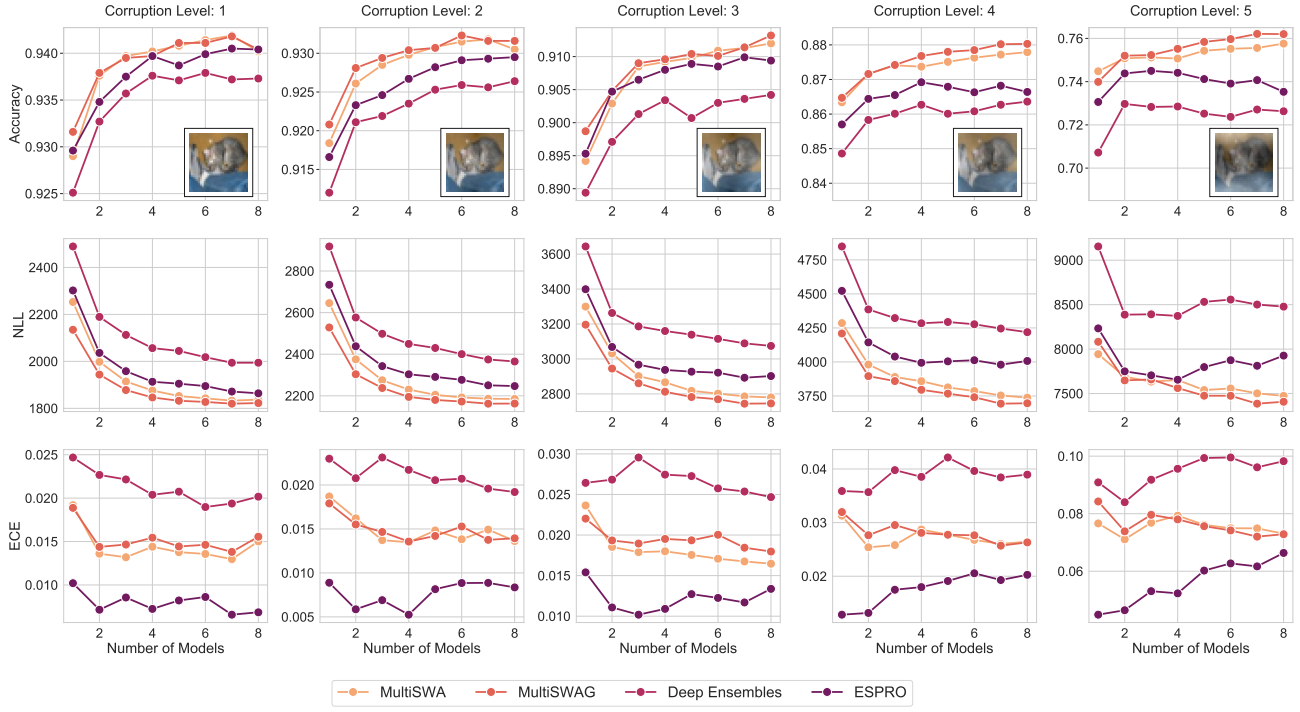


Figure A.10. Accuracy, NLL and ECE with increasing intensity of the *fog* corruption (from left to right).

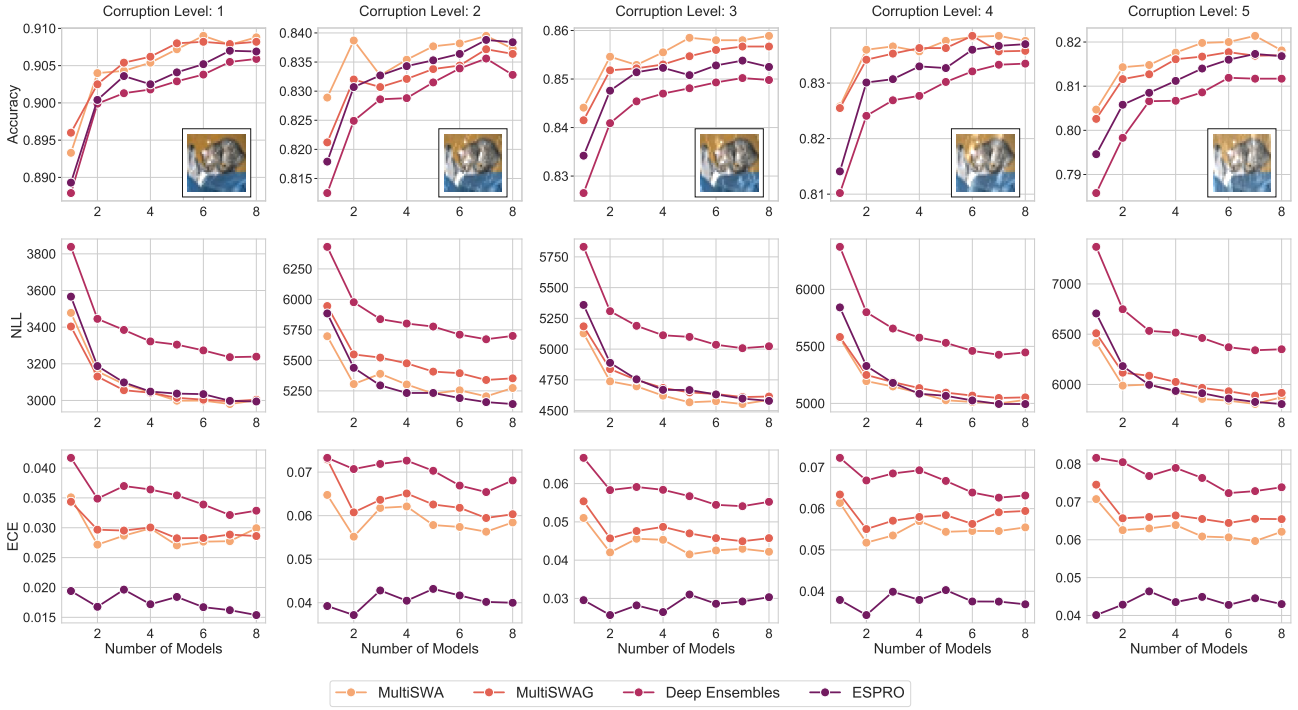


Figure A.11. Accuracy, NLL and ECE with increasing intensity of the *snow* corruption (from left to right).

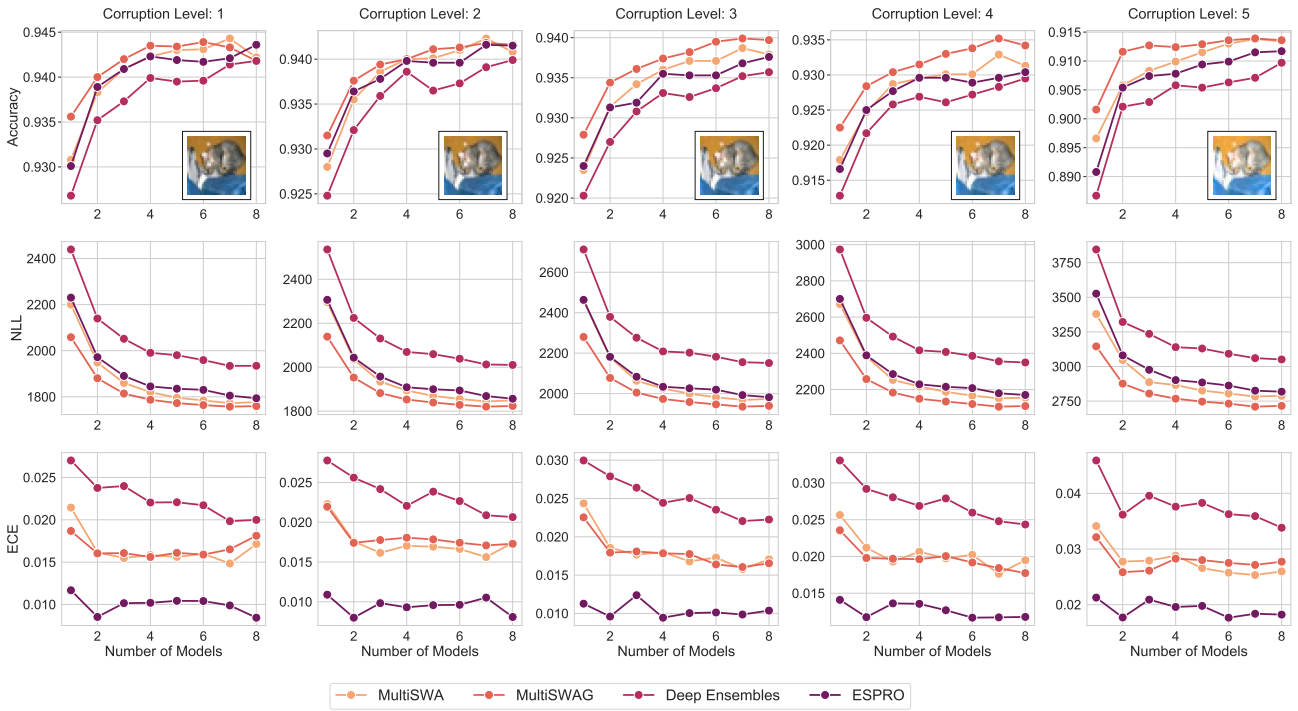


Figure A.12. Accuracy, NLL and ECE with increasing intensity of the *brightness* corruption (from left to right).

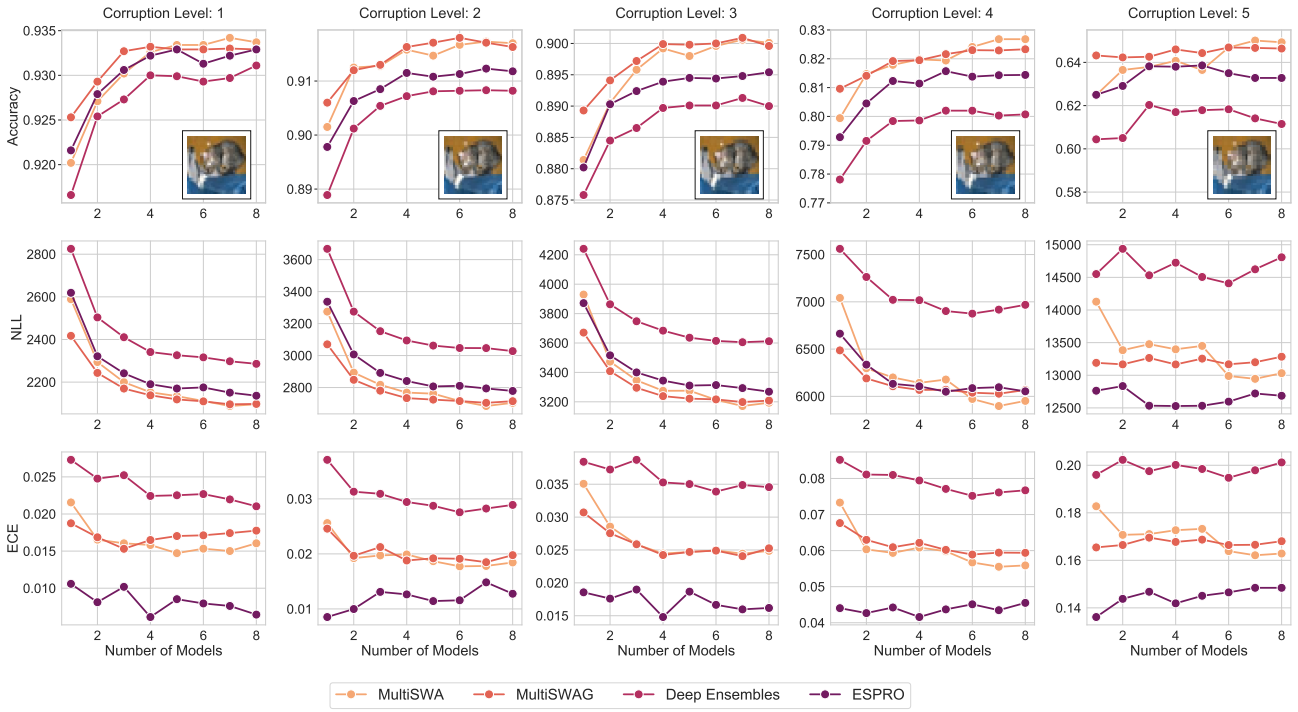


Figure A.13. Accuracy, NLL and ECE with increasing intensity of the *pixelate* corruption (from left to right).

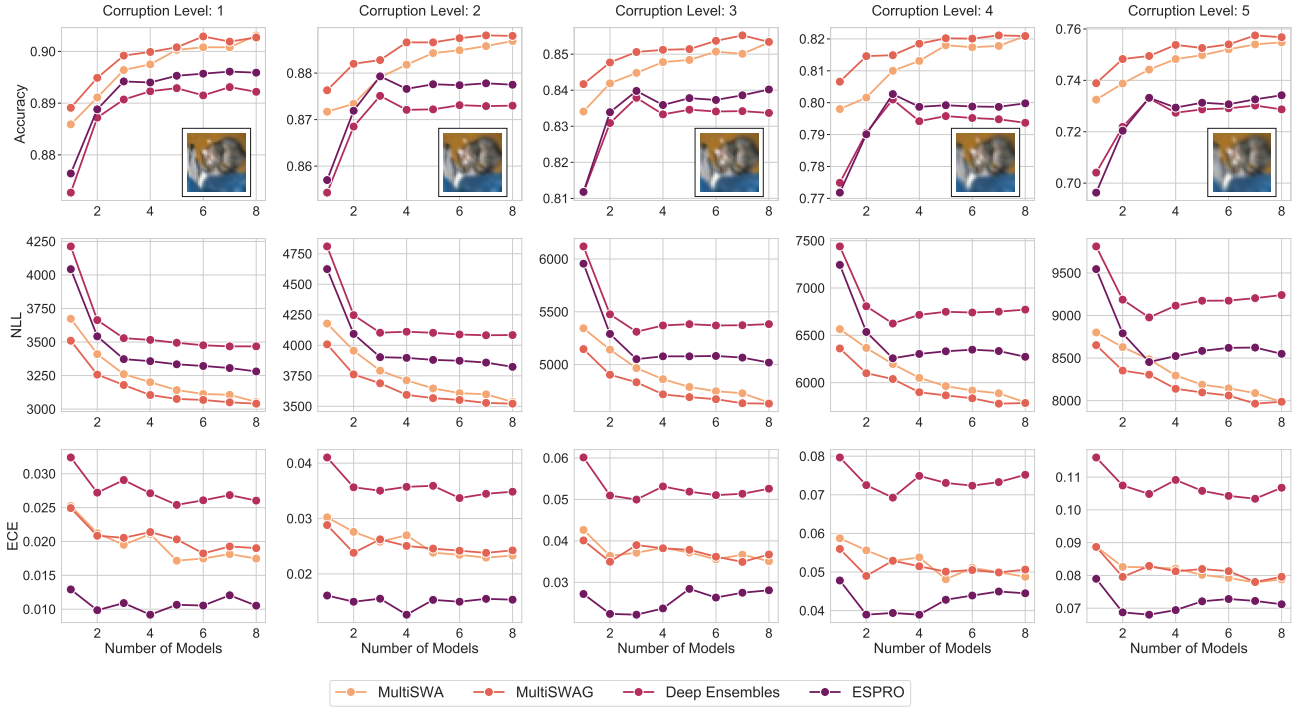


Figure A.14. Accuracy, NLL and ECE with increasing intensity of the *zoom blur* corruption (from left to right).

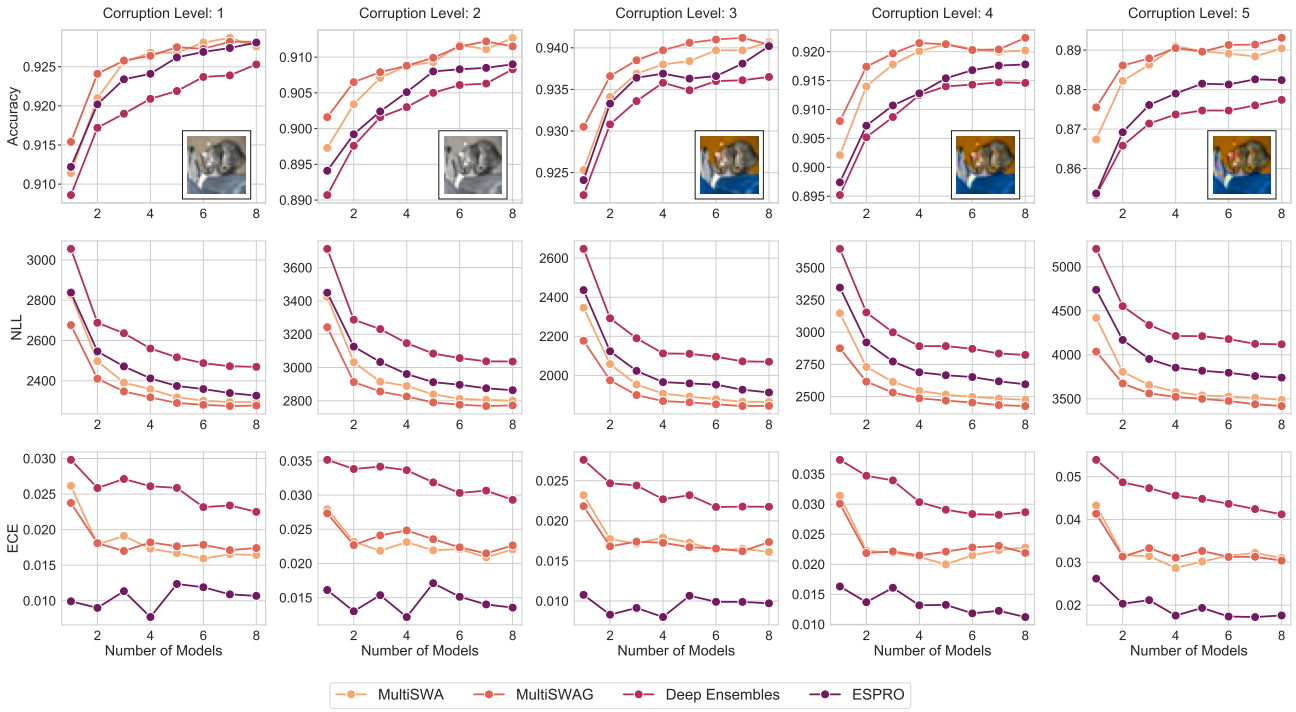


Figure A.15. Accuracy, NLL and ECE with increasing intensity of the *saturate* corruption (from left to right).

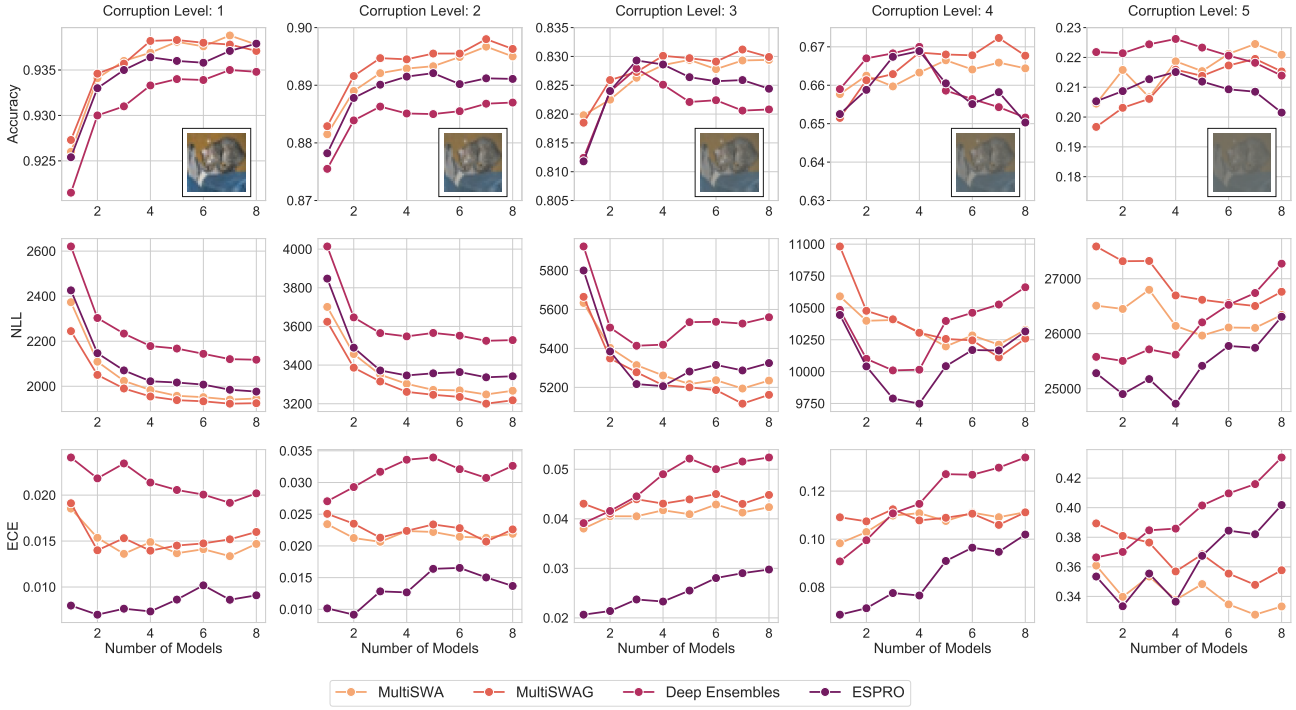


Figure A.16. Accuracy, NLL and ECE with increasing intensity of the *contrast* corruption (from left to right).

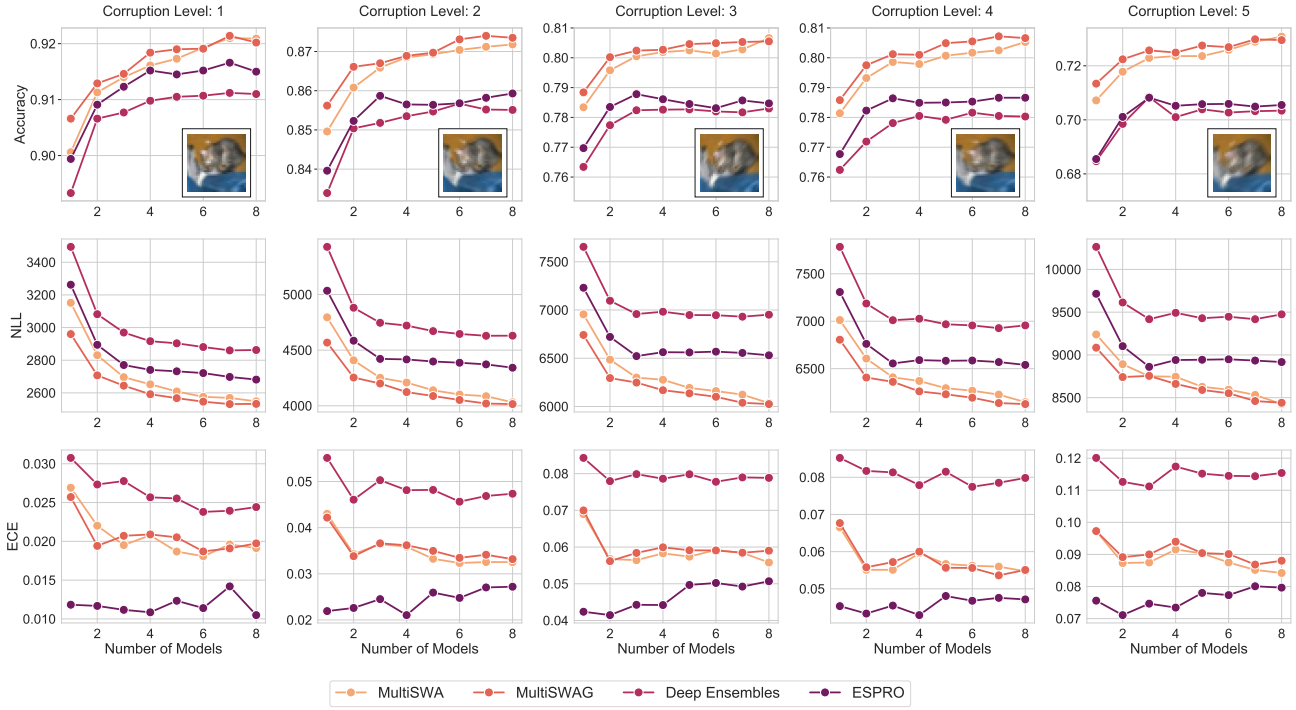


Figure A.17. Accuracy, NLL and ECE with increasing intensity of the *motion blur* corruption (from left to right).

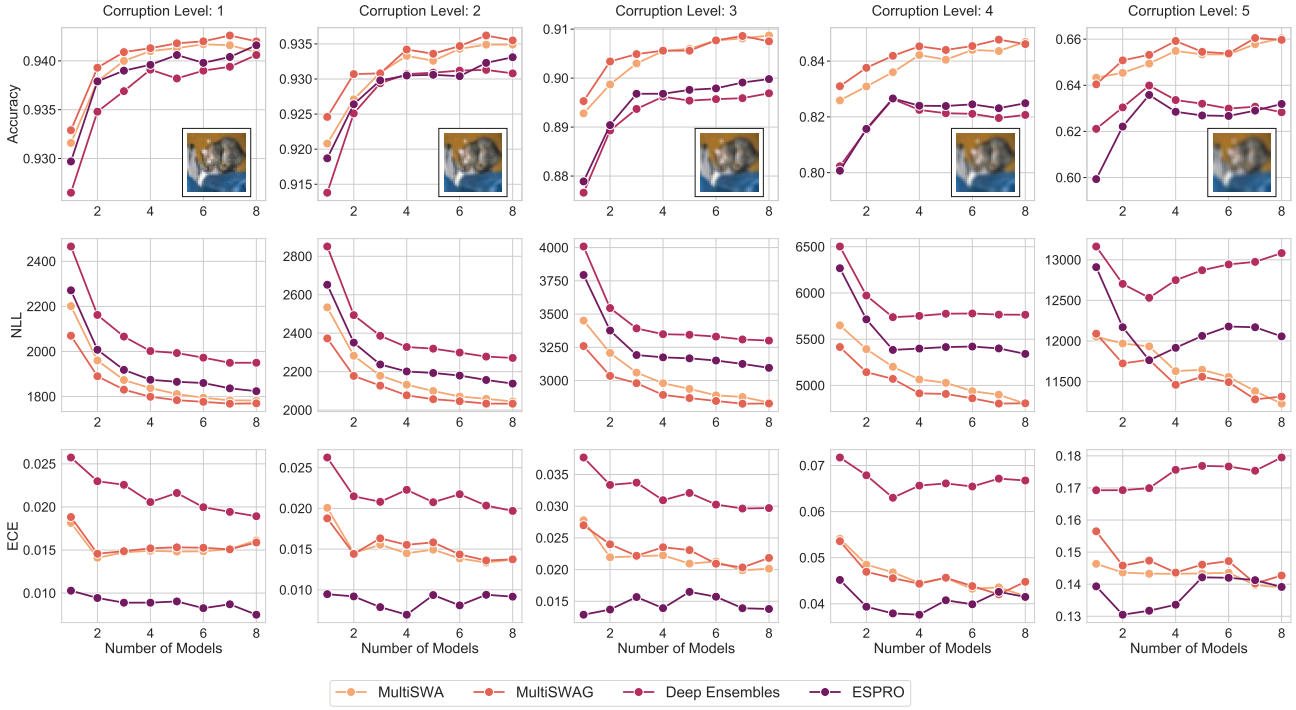


Figure A.18. Accuracy, NLL and ECE with increasing intensity of the *defocus blur* corruption (from left to right).

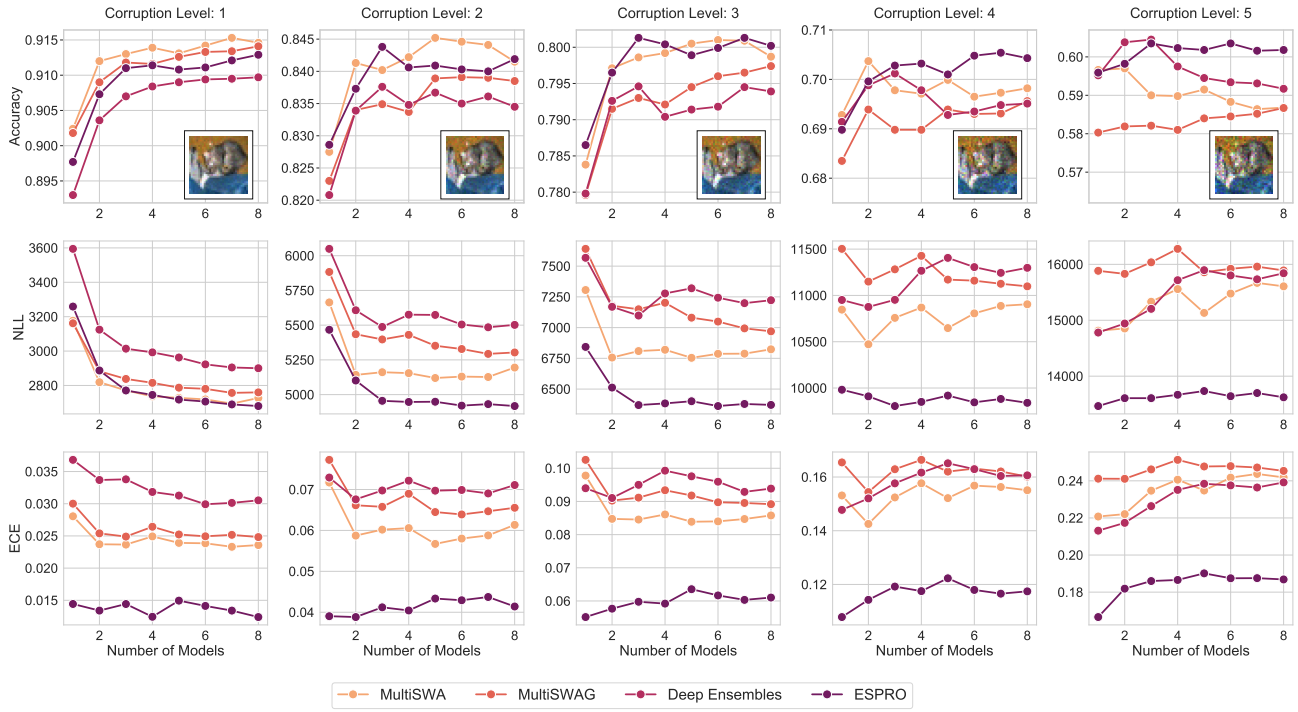


Figure A.19. Accuracy, NLL and ECE with increasing intensity of the *speckle noise* corruption (from left to right).

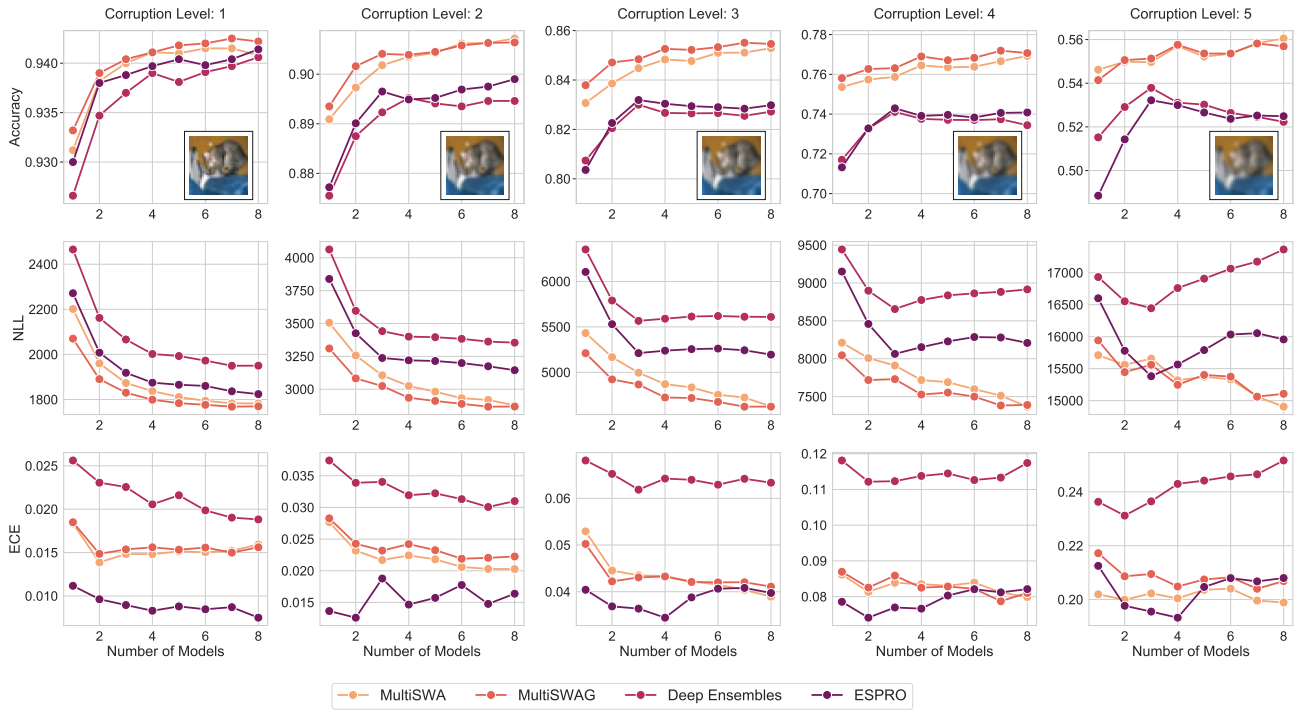


Figure A.20. Accuracy, NLL and ECE with increasing intensity of the *Gaussian blur* corruption (from left to right).

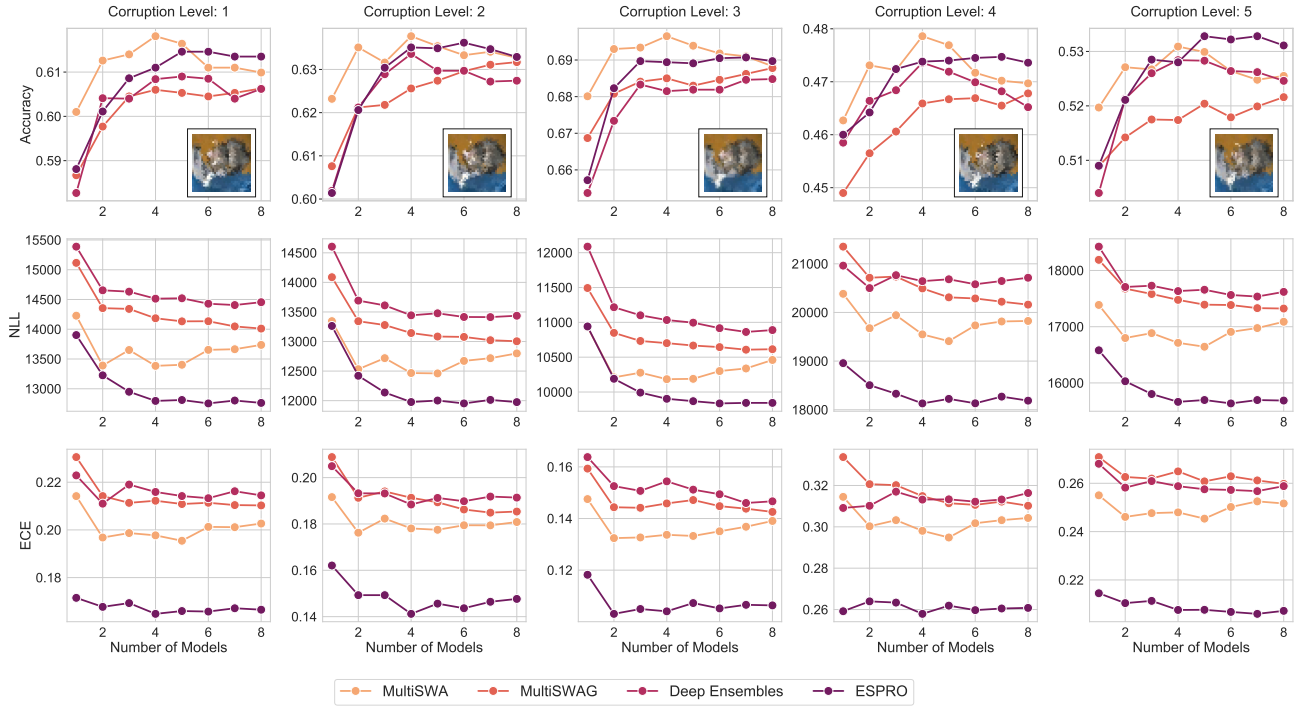


Figure A.21. Accuracy, NLL and ECE with increasing intensity of the *glass blur* corruption (from left to right).

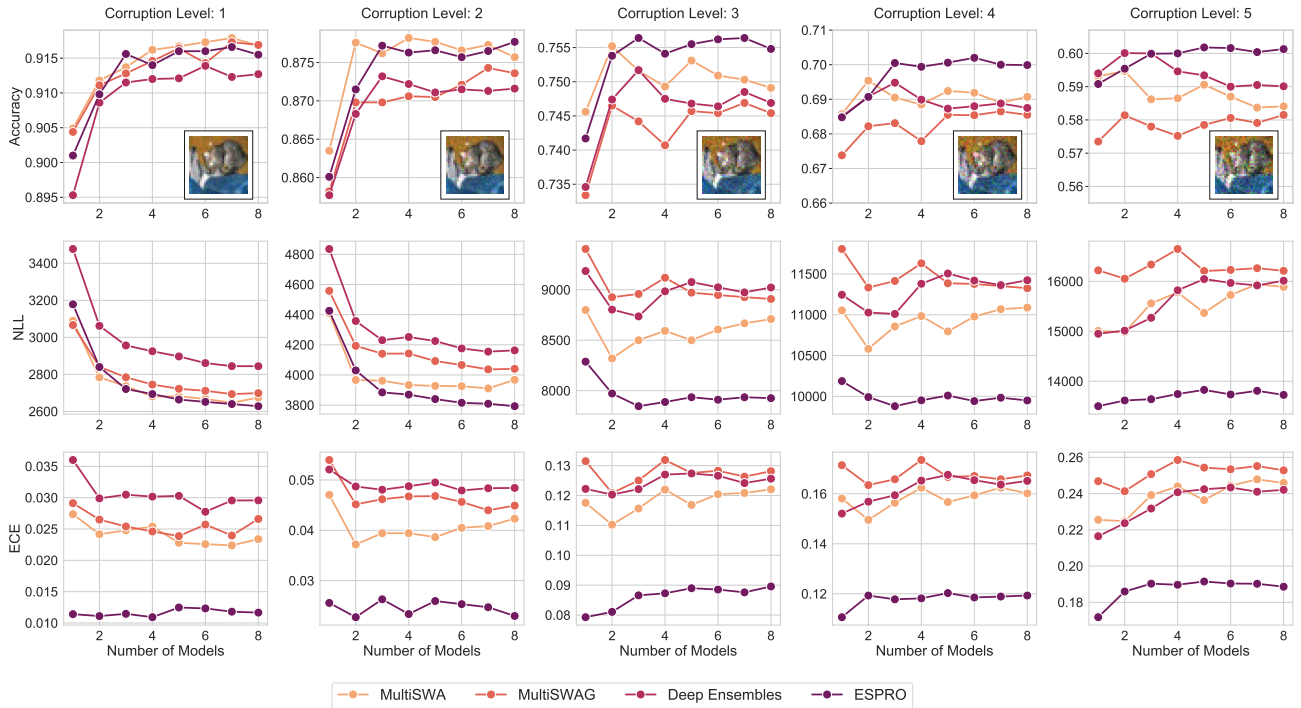


Figure A.22. Accuracy, NLL and ECE with increasing intensity of the *shot noise* corruption (from left to right).

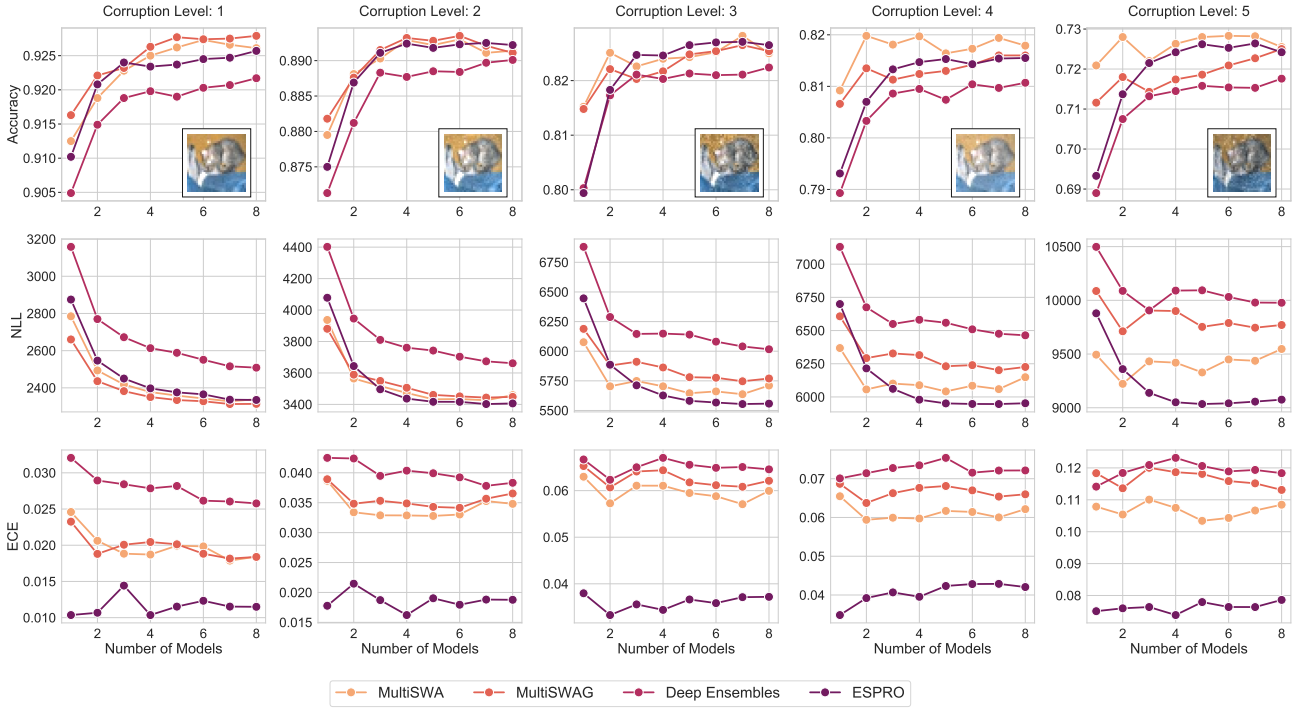


Figure A.23. Accuracy, NLL and ECE with increasing intensity of the *frost* corruption (from left to right).

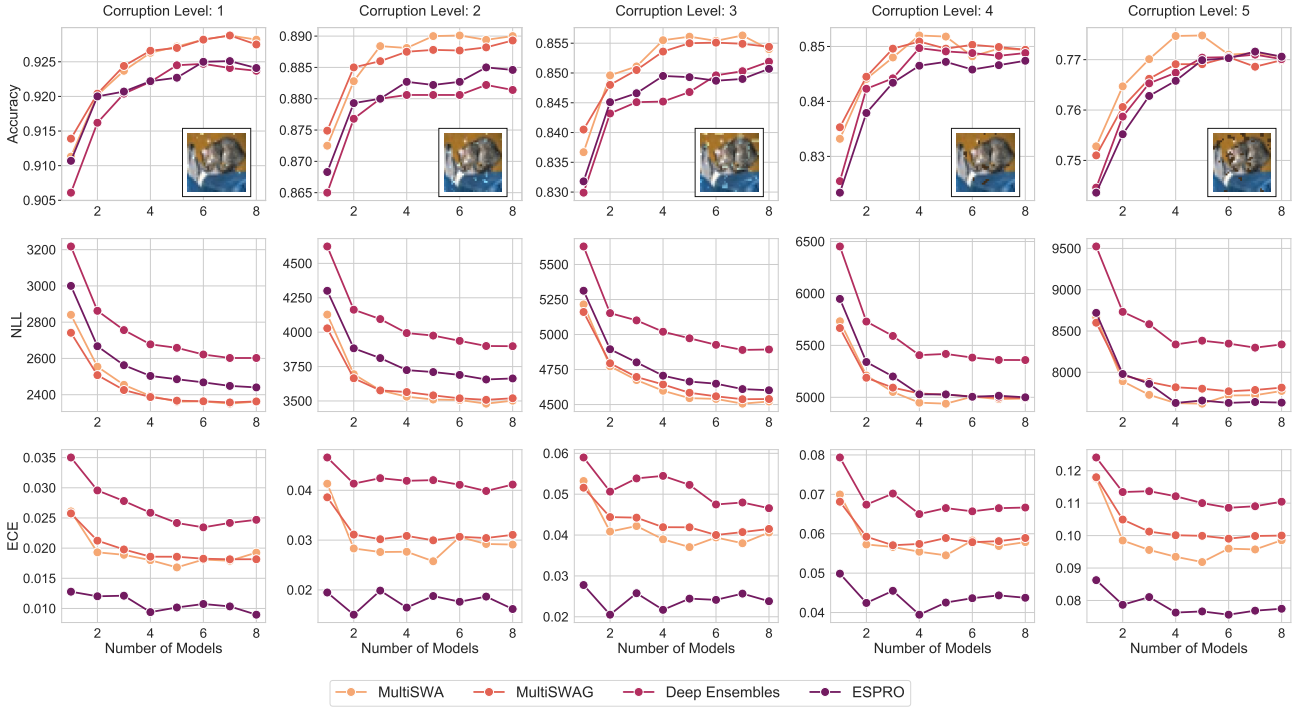


Figure A.24. Accuracy, NLL and ECE with increasing intensity of the *spatter* corruption (from left to right).

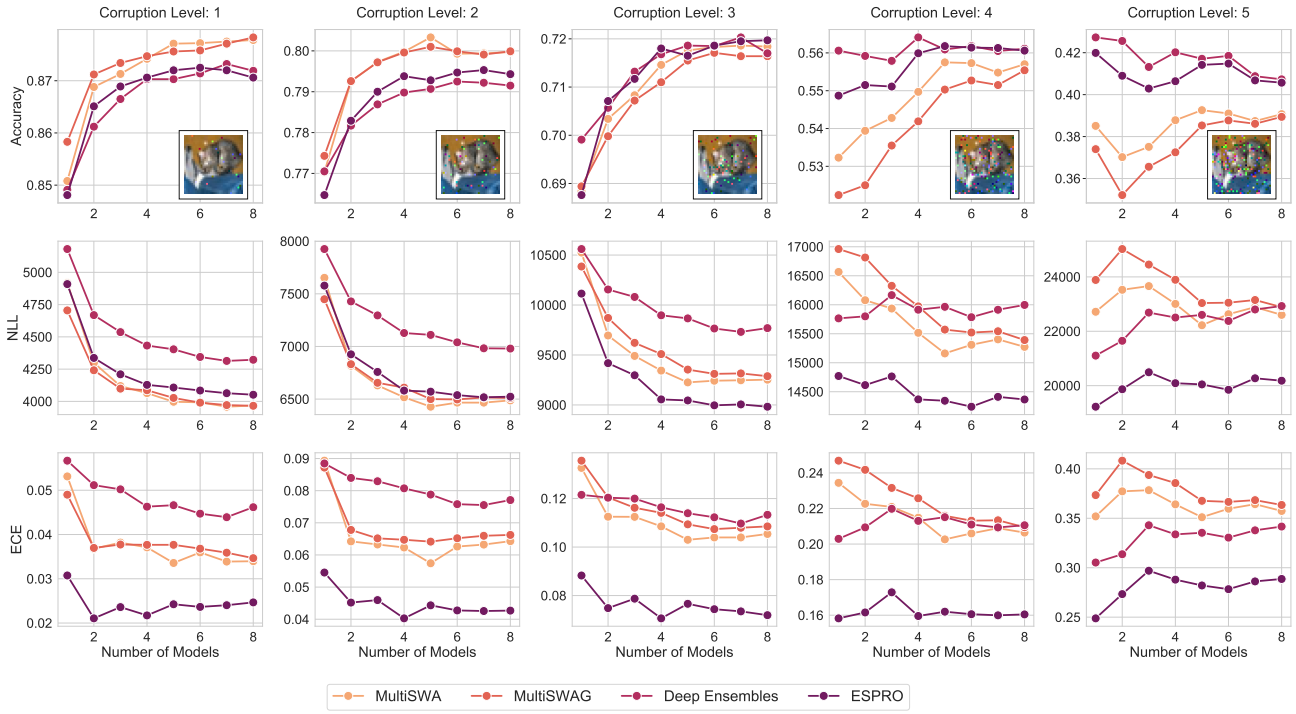


Figure A.25. Accuracy, NLL and ECE with increasing intensity of the *impulse noise* corruption (from left to right).

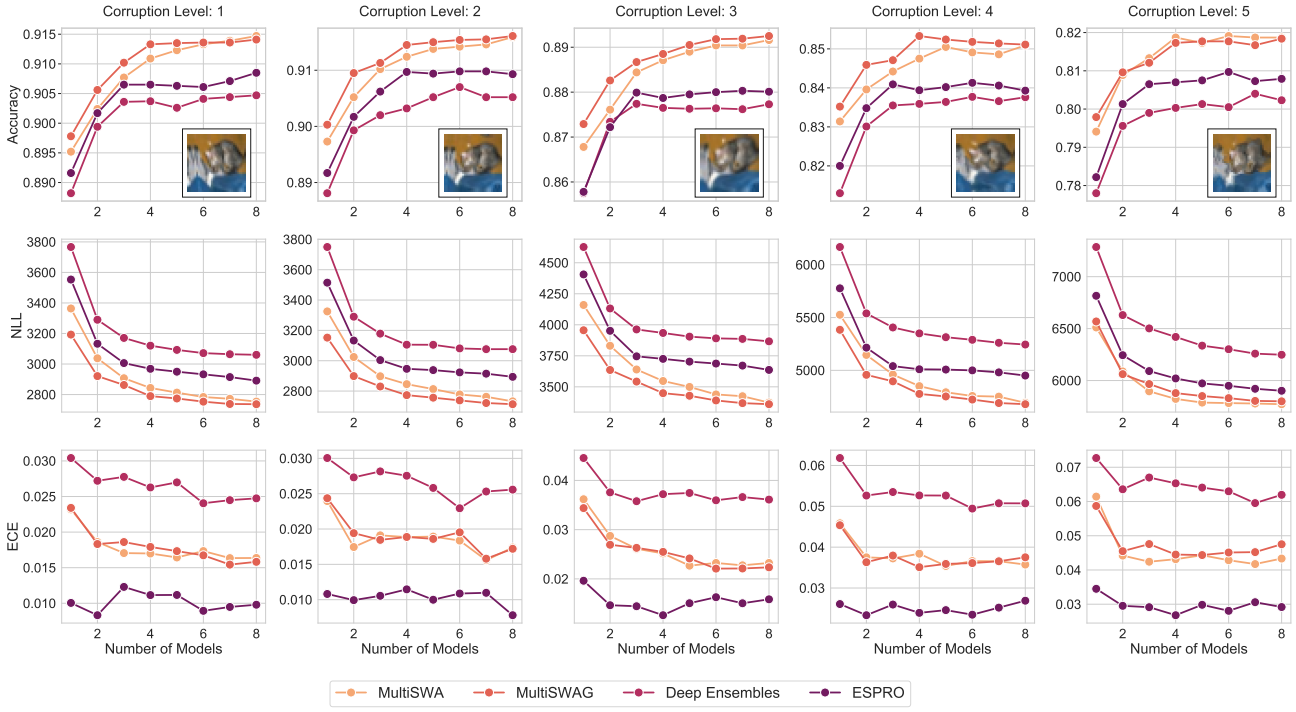


Figure A.26. Accuracy, NLL and ECE with increasing intensity of the *elastic transform* corruption (from left to right).