Stochastic Layers in Vision Transformers

Nikola Popovic\textsuperscript{1}, Danda Pani Paudel\textsuperscript{1}, Thomas Probst\textsuperscript{1}, Luc Van Gool\textsuperscript{1,2}
\textsuperscript{1}Computer Vision Laboratory, ETH Zurich, Switzerland
\textsuperscript{2}VISICS, ESAT/PSI, KU Leuven, Belgium
\{nipopovic, paudel, probstt, vangool\}@vision.ee.ethz.ch

Abstract

We introduce fully stochastic layers in vision transformers, without causing any severe drop in performance. The additional stochasticity boosts the robustness of visual features and strengthens privacy. In this process, linear layers with fully stochastic parameters are used, both during training and inference, to transform the feature activations of each multilayer perceptron. Such stochastic linear operations preserve the topological structure, formed by the set of tokens passing through the shared multilayer perceptron. This operation encourages the learning of the recognition task to rely on the topological structures of the tokens, instead of their values, which in turn offers the desired robustness and privacy of the visual features. In this paper, we use our features for three different applications, namely, adversarial robustness, network calibration, and feature privacy. Our features offer exciting results on those tasks. Furthermore, we showcase an experimental setup for federated and transfer learning, where the vision transformers with stochastic layers are again shown to be well behaved. Our source code will be made publicly available.

1. Introduction

Neural networks have shown to benefit from some level of stochasticity in various forms, including regularization\cite{82}, optimization\cite{39,79,80}, data augmentation\cite{15}, monte-carlo sampling for variational inference\cite{25,46,77,96} or uncertainty estimation\cite{25,26,43}, as well as gradient obfuscation for adversarial robustness\cite{65,70}. In general, it is however not clear how much and what form of stochasticity is desired for given neural architectures and for the tasks to be performed.

In this work, we are interested to systematically introduce and analyze the effect of stochastic layers in vision transformers (ViTs)\cite{20}. ViTs are networks of gripping interest, not only due to their widespread use but also due to their versatility. In turn, transformer architectures allow us to introduce stochastic layers in a controlled and tractable manner. This is facilitated by the token-wise processing nature, where the stochastic layer can perform a linear transformation, without altering the higher order topological structures represented by the image tokens.

The main idea of this paper is to introduce stochastic layers both during training and inference, within the existing architectures of visual transformers. These stochastic layers are introduced in an attempt to render the visual features robust and suitable for privacy, while maintaining the recognition performance of the initial visual transformers. The proposed stochastic layers operate as a linear function on tokens, and are introduced inside every multilayer perceptron of the transformer. These stochastic operations are performed on each channel dimension of every feature, thus making the layers fully stochastic in nature.

One important aspect of our way of injecting stochasticity is with regard to its consistency across the tokens. More precisely, we perform the same stochastic operation on all the tokens passing through a given multilayer perceptron. Such consistent linear operations preserve the topological structures invariant under affine transformations. Therefore, our way of injecting stochasticity encourages recognition to rely on the topological structures formed by the tokens, instead of their values. On the one hand, the topological structure based decisions make the visual transformers more robust to minor changes such as adversarial noise. On the other hand, the stochastic operations on the visual features makes them more private, when shared, as the recovery of the previous stage is particularly difficult due to the unknown stochastic operations. Our experiments show that the private feature extractor still preserves the information necessary to perform a variety of vision tasks. Our process of stochasticity injection tailored to vision transformers differs from existing ones\cite{65,77,82,96} in one or more of these three aspects: computational, tractability, or performance. The details are discussed in Section 3 and 5.

In this paper, we examine our stochastic layers with regards to adversarial robustness, network calibration, and...
Figure 1. **An overview of the effects of proposed stochastic layers in vision transformers.** The channel $j$ of every token inside the transformer’s MLP block is multiplied with the same random variable $\alpha^j \sim \mathcal{U}(1 - \Delta, 1 + \Delta)$ from (8). In 1a we see that there is no significant drop in accuracy, even when using strong stochasticity during inference. Next, in 1b we see that adding stochastic layers can improve network calibration, meaning that the predictive probability better depicts the uncertainty. In 1c we can see that adding stochastic layers can improve robustness against adversarial attacks. Finally, in 1d we see that adding stochastic layers can help with privacy preservation by reducing reconstruction quality of feature maps sent over unsecure channels.

feature privacy. Our features offer exciting results for those. Furthermore, we also showcase experimental setups for federated and transfer learning, where the vision transformers with stochastic layers are again shown to be well behaved.

2. Related Work

In what follows, we present related work with regards to the transformer architecture and stochasticity on a high level. For a detailed discussion of the literature on the issues of adversarial robustness, private federated learning, and network calibration, please refer to Section 5.

**Vision Transformer.** Based on the self-attention mechanism [87], the transformer was first introduced in natural language processing [17, 18, 87, 94]. In order to capture context across input sequences, the transformer employs multi-head self-attention and multi-layer perceptron modules, allowing for global interaction. Through the seminal work of ViT [20], transformers have successfully entered the vision domain by dividing an image into patches that are treated as a sequence. Vision transformers achieve state-of-the-art performance on problems including image classification [90, 97], object detection [9], semantic/instance segmentation [101], and video segmentation [91]. For a more complete survey of vision transformers, we refer to [44].

**Stochasticity in Neural Networks.** Stochasticity has been a subject of study with regards to its effects on regularization and optimization [62, 79, 82], data augmentation [15, 54], monte-carlo sampling for the variational inference of latent variables [46] and model parameters [25, 45], as well as generative modelling [42, 46, 60]. Furthermore, noise injection has been successfully applied to uncertainty estimation and network calibration [25, 26, 43, 52], adversarial robustness [1, 13, 70], and network compression [16].

If people do not believe that mathematics is simple, it is only because they do not realize how complicated life is.

john von Neumann (1903-1957)
of the performed operations is shown in Figure 2.

Figure 2. A single stochastic multilayer perceptron with multiple input tokens \( T \). At the intermediate stage, the stochastic layer \( f(x) \) homeomorphically maps features \( X \) to \( Y \).

**Inside the multilayer perceptron in ViTs:** ViTs divide images into multiple tokens. These tokens are processed by attention modules followed by multilayer perceptrons (MLPs). We are interested in introducing stochasticity within the processing pipeline of the MLPs. Let us consider, without lack of generality, that the intermediate features inside an MLP for tokens \( T = \{ t_i \}_{i=1}^n \) are given by \( X' = \{ x_i \}_{i=1}^n \in \mathbb{R}^d \). A homeomorphic mapping \( f : X \rightarrow Y \), in a fully stochastic fashion, is applied to the set of features \( X' \) resulting into the mapped set \( Y \). We continue processing the mapped features \( Y' \) similar to \( X' \), as if the stochastic layer represented by \( f \) was absent. We represent \( f \) as a square diagonal matrix \( A \in \mathbb{R}^{d \times d} \) with non-zero diagonal entries. Then the operation by the introduced stochastic layer in the form, \( y_i = f(A, x_i) = Ax_i, \forall x_i \in X', \) ensures \( f : X \rightarrow Y \) to be homeomorphic. Please note that the mappings by square diagonal matrices with non-zero entries are continuous with their inverse being continuous as well. Other choices of \( f \) could be made as well, but we study the behaviour of ViTs with a relatively simple function \( f \).

4. Stochastic Layers in Vision Transformers

We now describe in detail how our stochastic layer is employed inside a vision transformer. Let the input image be denoted as \( I \in \mathbb{R}^{H \times W \times 3} \), where \( H \) is the height and \( W \) is the width of the image. The image is represented using 3 color channels. A feature extractor \( F \) takes the image \( I \) as the input and produces a feature map \( F = F(I) \). Usually \( F \in \mathbb{R}^{\frac{H}{T} \times \frac{W}{T} \times C} \), meaning that its spatial dimensions are downscaled by a factor of \( s \) compared to the image \( I \) and that it has \( C \) channels. The feature map \( F \) can be used for various different tasks. As one option, the feature map can be given as an input to a classification model \( C \) to produce \( O = C(F) = C(F(I)) \), where \( O \in \mathbb{R}^m \) and \( m \) is the number of classes. It can also be given to a regression model \( R \) to produce \( O = R(F) \), where \( O \in \mathbb{R}^m \) and \( m \) is the number of regressed values. Another option is to give it to a model \( D \) to produce an image-to-image translation output \( O = D(F) \), where \( O \in \mathbb{R}^{H \times W \times m} \). This can be used for various tasks like semantic segmentation, depth estimation, flow estimation, etc.

Generally speaking, a Vision Transformer (ViT) [20] is one type of feature extractor \( F \). First, it takes the input image \( I \) and divides it spatially into \( k \times k \) patches to obtain \( \tilde{I} \in \mathbb{R}^{\frac{H}{k} \times \frac{W}{k} \times 3k^2} \), which represents \( n_T = \frac{n}{k^2} \) token vectors of dimension \( 3k^2 \). A popular choice is \( k = 16 \) [20, 85]. Then, each token vector is embedded into \( d_T \) dimensions, using the same affine transformation, to obtain \( \mathcal{F}_0 \in \mathbb{R}^{n_T \times d_T} \). Optionally, a classification or regression token can be appended to \( \mathcal{F}_0 \), which would result in \( n_T = \frac{n}{k^2} + 1 \) . This token is used to make a prediction for the final task. Next, a positional encoding \( P \in \mathbb{R}^{n_T \times d_T} \) is added to \( \mathcal{F}_0 \) which gives \( \mathcal{F}_1 = \mathcal{F}_0 + P \). The purpose of the positional encoding \( P \) is to add information about which token in \( \mathcal{F}_0 \) corresponds to which spatial location of the input image \( I \). The transformer then uses \( l \) consecutive transformer blocks \( B_i \) to extract features. Each block \( B_i \) takes the output of the previous block \( \mathcal{F}_{i-1} \) as its input and produces \( \mathcal{F}_i = B_i(\mathcal{F}_{i-1}) \), where \( \mathcal{F}_i \in \mathbb{R}^{n_T \times d_T} \). The first block takes \( \mathcal{F}_0 \) as input. To obtain the final feature map \( \mathcal{F} \), we usually take the output of the last block \( \mathcal{F}_l \), discard the optional classification or regression token, and spatially rearrange it back as \( \mathcal{F} \in \mathbb{R}^{\frac{H}{T} \times \frac{W}{T} \times C} \), where \( C = d \) and \( s = k \).

Each transformer block \( B_i \) is composed of a Multi-head Self Attention block (MSA), followed by a Multilayer Perceptron (MLP) [20, 87]. Self-attention is a spatially global operation, where every token interacts with every other token and thus information is shared across spatial dimensions. Multiple heads in the MSA block are used for more efficient computation and for extracting more diverse features. The MSA block executes the following operations:

\[
Z_i = \text{MSA}(\text{LN} (\mathcal{F}_{i-1})) + \mathcal{F}_{i-1},
\]

where LN represents a Layer Normalization [3]. The MSA block is followed by the MLP block, which processes each token separately using the same Multilayer Perceptron. This block processes the token features, after their spatially global interactions in the MSA block, by sharing and refining the representations of each token across all of its channels. The MLP block executes the following operations:

\[
\mathcal{F}_i = \text{MLP}(\text{LN}(Z_i)) + Z_i, \quad Z_i \in \mathbb{R}^{n_T \times d_T}.
\]

In the following, we omit the block index \( i \) for simplicity. Unpacking (2) corresponds to the operations:

\[
Z_{i}^{\text{LN}} = \text{LN}(Z), \quad Z_{i}^{\text{LN}} \in \mathbb{R}^{n_T \times d_T},
\]

\[
Z_{i}^{\text{FC}_1} = \text{FC}(Z_{i}^{\text{LN}}), \quad Z_{i}^{\text{FC}_1} \in \mathbb{R}^{n_T \times d},
\]

\[
Z_{i}^{\sigma} = \sigma(Z_{i}^{\text{FC}_1}), \quad Z_{i}^{\sigma} \in \mathbb{R}^{n_T \times d},
\]

\[
Z_{i}^{\text{FC}_2} = \text{FC}(Z_{i}^{\sigma}), \quad Z_{i}^{\text{FC}_2} \in \mathbb{R}^{n_T \times d_T},
\]

\[
\mathcal{F}_i = Z_{i}^{\text{FC}_2} + Z_i, \quad \mathcal{F}_i \in \mathbb{R}^{n_T \times d_T},
\]

with fully connected layers FC and activation function \( \sigma \).
5. Experiments

In this section, we analyze the effect of our linear stochastic layers for vision transformers on several aspects, including accuracy, adversarial robustness, confidence calibration, and privacy. Furthermore, we showcase applications for federated and transfer learning, considering several image understanding tasks. For this purpose, we will use a DeiT-S transformer architecture [85] in all our experiments. DeiT-S has a parameter count and computational complexity similar to ResNet-50 [34].

Training Procedure. To explore the general behaviour of our method we use the ILSVRC-2012 ImageNet-1k dataset, which contains 1.2M training and 50000 validation images grouped into 1000 classes [73]. The DeiT-S network pre-trained on ImageNet-1k is used as a starting point. All the details regarding the pre-training procedure are contained in [85]. We insert the stochastic layer from (8) into every block of the DeiT-S and fine-tune for 30 epochs. We choose the distribution $\mathcal{P}$ as uniform $\mathcal{P} \sim \mathcal{U}(1 - \Delta, 1 + \Delta)$. During this fine-tuning process we use the AdamW optimizer [56] and a cosine scheduler with a learning rate of $10^{-5}$, which gradually decays to $10^{-6}$. We also turn off DropPath [36] and use the ReLU activation in order to introduce the noise to a deterministic baseline with a (piecewise) linear impact. All other settings remain as in pre-training.

5.1. Effect on ImageNet Accuracy

We begin by analyzing the predictive performance of stochastic layers with regards to classification accuracy on the ImageNet validation set after fine-tuning. As presented in Figure 1a, there is no significant drop in accuracy, even when using high noise levels during inference. The network performs better if $a^l$ is set to the mean of the training noise distribution $E[\mathcal{P}]$. The network also improves when multiple predictions are sampled ($a^l \sim \mathcal{P}$, independently sampled for each block and prediction) and averaged in a Monte Carlo fashion (see Section 5.2.2 for details). However, the inference also works reasonably well when we just sample once. This is surprising given the fact that the random variables $a^l$, which multiplies token channels, is sampled from a completely uninformative distribution. Which noise strength and inference mode to choose will depend on the final use-case. In the following experiments, we analyse the involved trade-offs to guide this decision.

5.2. Confidence Calibration

In practice, any inference algorithm is only useful for a decision-making system, if it can also give an indication about the confidence of its predictions. Besides good accuracy, estimating the confidence is crucial for real-world applications, where decisions have to be made with incomplete information. Intuitively, if a well-calibrated classifier outputs a prediction with a confidence of $p$ (e.g. softmax score), we expect the classifier to be correct with probability $p$. It has been observed that training of complex neural networks may result in bad calibration, despite exhibiting good accuracy. Different factors, including model capacity, architecture, regularization and loss function influence the calibration quality [31, 59]. In the following, we introduce the calibration problem in more detail, and then analyse the role of stochasticity through our experiments.

5.2.1 Problem Definition

We define the confidence calibration problem in the context of supervised multi-class classification, following [31]. Given the random variables $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$, representing the input and labels respectively. Then a classifier...
\( h(X) = (\hat{Y}, \hat{P}) \), predicting a label \( \hat{Y} \) and an associated confidence \( \hat{P} \), would be perfectly calibrated if

\[
\mathbb{P}(\hat{Y} = Y|\hat{P} = p) = p \ \forall p \in [0, 1].
\] (9)

In particular, we are interested in quantifying the quality of calibration. To this end, we analyse the expected calibration error (ECE),

\[
\mathbb{E}_P\left[ \mathbb{P}(\hat{Y} = Y|\hat{P} = p) - p \right].
\] (10)

ECE measures the expected disparity between prediction confidence and accuracy, and would be 0 in the case of (9).

### 5.2.2 Monte Carlo Inference

To achieve a good calibration, we aim to minimize the ECE (10). It has been demonstrated in the literature that model uncertainty (i.e. the posterior distribution on the network weights \( w \)) can be estimated using monte-carlo sampling in the context of dropout [25, 26, 43]. This way, predictions are integrated implicitly over the posterior distribution of the network weights \( q(w) \) as,

\[
p(y|x) \approx \int p(y|x, w)q(w)dw \approx \frac{1}{T} \sum_{t=1}^{T} p(y|x, \hat{w}_t),
\] (11)

approximated by \( T \) samples \( \hat{w}_t \) [25], generated by applying dropout at test time. Since our stochastic layers naturally offer such sampling for vision transformers, our goal is to investigate their effect on network calibration when performing inference with (11).

### 5.2.3 Calibration Experiments

#### Evaluation Metric

To estimate (10) using finite samples, [61] sorts network predictions \( \hat{P} \) into \( M \) equidistant bins. With \( B_m \) denoting the indices of all samples in the interval \((\frac{x_m-1}{N}, \frac{x_m}{N}]\), we compute the accuracy and confidence for each bin as,

\[
\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} 1(\hat{y}_i = y_i),
\] (12)

\[
\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i.
\] (13)

Then the ECE is approximated by computing the disparity between accuracy and confidence for samples from each bin as,

\[
\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)|.
\] (14)

For our experiments, we compare the ECE with \( M = 15 \) bins [31] on the ImageNet-1k validation set.

### Table 1. Confidence calibration. Stochastic layers improve the confidence calibration of Vision Transformers compared to the strong Dropout baseline. The calibration is better for stronger noise, and when using Monte Carlo sampling.

|                  | \( \Delta = 0 \) | \( N = 1 \) | \( N = 50 \) |
|------------------|------------------|------------|------------|
| Regular (without temperature) | 0.087 | 0.087 | 0.087 |
| Regular          | 0.025 | 0.025 | 0.025 |
| Training Dropout | 0.022 | 0.023 | 0.019 |
| \( \Delta = 0.5 \) Ours | 0.023 | 0.020 | 0.015 |
| Training Dropout | 0.019 | 0.016 | 0.014 |
| \( \Delta = 1.0 \) Ours | 0.016 | 0.013 | 0.011 |

#### Setup

We evaluate the calibration of vision transformers with and without stochastic layers, using models from Section 5.1. As a baseline we compare to our deterministic model, as well as to Monte Carlo dropout [26], which is a strong baseline for uncertainty estimation. To this end, we replace our stochastic layers with dropout. To create a fair counterpart for our model with a noise level \( \Delta \), we choose the dropout probability \( p \) such that its underlying Bernoulli distribution matches the variance of the distribution \( P \) from (8). The strongest baseline would be to use model ensembles [51]. However, we do not include ensembles due to the computational demand on big datasets, which also limits the practical application. On the other hand, a very simple and effective technique to improve confidence calibration is given by softmax temperature tuning [31]. Very recently, this has been also shown to work well with transformer architectures [59]. We use it in our experiments, since it can be simply applied to the baselines and our models during evaluation.

#### Results

The results are summarized in Table 1. Compared to the baseline, temperature scaling [31] improves the calibration quality. The best results however are achieved in combination with our approach, through Monte Carlo sampling and confidence estimation. Calibration accuracy tends to improve with higher number of samples.

### 5.3. Adversarial Robustness

Besides the accuracy and privacy, another important aspect of the network is its behaviour with respect to adversarial samples [83]. If we are interested to design algorithms that use visual features which resemble the human perception, then small adversarial perturbations in the pixel intensities of the input image must not lead to different decisions in the output [37]. Also, we want to avoid exposition of such vulnerabilities to actors with malicious intent. We therefore desire networks that are adversarially robust [28].

#### Adversarial Attacks

In general, adversarial attacks exploit the differentiability of the network \( h(x) \) and its prediction loss \( L(h(x)) \) with respect to the input image \( x \). Given
Table 2. Adversarial robustness. Stochastic layers increase adversarial robustness in Vision Transformers. Combining the effects of strong noise with Monte Carlo sampling can improve adversarial robustness and clean accuracy under adversarial training, as shown in Table 2a. Without adversarial training (see Table 2b), stochastic layers offer some level of adversarial robustness with a slight drop in accuracy, compared to the non-robust baseline.

(a) With adversarial training

| Accuracy ↑ | Adversarial training with $\epsilon = 2$ | Adversarial training with $\epsilon = 4$ |
|------------|---------------------------------------|---------------------------------------|
| Clean samples | PDG–10 attack | Clean samples | PDG–10 attack |
| Regular | 71.61% | 42.33% | 65.01% | 27.24% |
| Training $\Delta = 0.1$ | 71.79% | 42.90% | 65.25% | 27.88% |
| $N = 50$ | 71.84% | 42.91% | 65.29% | 27.86% |
| Training $\Delta = 0.5$ | 71.79% | 44.51% | 66.53% | 30.68% |
| $N = 50$ | 73.75% | 46.21% | 68.85% | 31.47% |
| Training $\Delta = 1.0$ | 68.62% | 45.37% | 65.40% | 33.48% |
| $N = 50$ | 73.83% | 49.32% | 70.47% | 36.38% |

(b) Without adversarial training

| Accuracy ↑ (Without adversarial training) | Clean samples | PDG–10 attack |
|---------------------------------------|----------------|----------------|
| Regular | 71.61% | 0.43% | 0.01% |
| Training $\Delta = 0.5$ | $N = 1$ | 77.43% | 5.25% | 1.41% |
| $N = 50$ | 70.72% | 5.25% | 1.37% |
| Training $\Delta = 1.0$ | $N = 1$ | 72.93% | 12.7% | 4.77% |
| $N = 50$ | 77.75% | 12.36% | 4.76% |

5.3.1 Robustness Experiments

Setup. In order to evaluate adversarial robustness, we craft adversarial examples for each image in the ImageNet-1k validation set, and test the model’s accuracy on those adversarial examples. We do this by using the PGD attack (16) with 10 iterations (PGD-10). When crafting each adversarial example, we initialize the attack’s starting point randomly inside the $\epsilon$-hypercube and restart this procedure 5 times to find the strongest attack, following the standard benchmark of [76, 92]. We test for two different $\epsilon$-hypercube constraints $\epsilon = \{2, 4\}$. The attack step $\alpha$ from (16) is 1 in both cases.

Results with Adversarial Training. First, we test the effect of our stochastic layers combined with adversarial robustness training. We take the same DeiT-S starting point as described in Section 5.1 and additionally use adversarial training while fine-tuning the inserted stochastic layers. More specifically, we use the efficient adversarial training described in [92]. Adversarial training of [92] is an effective remedy for adversarial vulnerability, and in fits in our standard training protocol [2, 57] without much overhead. The baseline is a deterministic model fine-tuned with adversarial training. During this fine-tuning process regular hyperparameters remain the same as in Section 5.1, except for using 20 epochs now. The adversarial training hyperparameters are as above, except for the step sizes which are $\alpha = \{2.5, 5.0\}$ for $\epsilon = \{2, 4\}$ respectively. In Table 2a we see that adding stochastic layers improves adversarial robustness beyond adversarial training. This effect is stronger if the networks are sampled in a Monte Carlo fashion. Also, while sampling, the clean accuracy becomes higher than in the baseline. This is particularly interesting, since it has been shown that there is a trade-off between adversarial robustness and clean accuracy [99].

Results without Adversarial Training. We also test the robustness of our stochastic layers without adversarial training. For this experiment we take the models trained in Section 5.1 and evaluate the robustness. In Table 2b we first notice that the baseline has no adversarial robustness, like it has been observed in literature [76]. We also notice that our stochastic layer achieves a significant level of robustness, even without adversarial training. Robustness becomes stronger as the strength of the stochasticity increases. We also observe that clean accuracy in our method can be largely recovered through Monte Carlo inference, especially in the case of high noise. For $\Delta = 1.0$, the clean accuracy improves from 72.93% $\rightarrow$ 77.75%.

5.4. Privacy-preserving Features

Sharing image features is an important part of collaborative systems that distribute computations across nodes. Applications include camera localization [81], structure-from-motion [27] in a shared environment, federated learning [6,
41, 47], or split learning/inference [32, 88]. To address privacy issues, problem-specific solutions have been proposed. [27, 81] obfuscate the geometry to maintain privacy of sparse features [68]. Homomorphic encryption guarantees privacy in the context of feature extraction [35, 69], retrieval [21], and federated learning [22]. However, these methods do not readily extend to many practical cases of distributed learning. This includes split learning [32, 88] where features of the data are shared after being processed by a part of the complete network. Even though the feature extraction removes information from the input image, it is still possible reconstruct the input [86, 89, 95, 100] or privacy-related attributes thereof [4, 78].

In Section 5.4.1 we first analyse to which extent vision transformers preserve the privacy of features. To this end, we train a decoder to reconstruct the input from features of different transformer blocks, and show the impact of our stochastic layers on reconstruction quality, and quantify the trade-off between accuracy and noise level. In Section 5.4.2 we further demonstrate that the stochastic network still retains its generalization ability and flexibility to transfer to other datasets, and is therefore a directly applicable for federated learning, while ensuring a higher level of privacy.

5.4.1 Input Image Reconstruction

We now analyse the models introduced in Section 5.1 with regards to feature privacy. To simulate a distributed learning environment, we split the model into two parts. The local network is the part of the network which processes the input up until the feature map $\tilde{Z}^{FC_i}$ from (8) in block $i$. It then sends $\tilde{Z}^{FC_i}$ to the centralized server where it is processed by the second part of the network.

**Threat Model.** The adversary intercepts feature maps $\tilde{Z}^{FC_i}$, while they are being sent to the server. Similarly, an adversary could potentially obtain a small amount of input images corresponding to the sent feature maps and use them to learn a model which reconstructs the input image. When a new feature map $\tilde{Z}^{FC_i}$ is sent to the server, the adversary is now able to reconstruct the corresponding input image and thus violate the users privacy.

**Decoder Attack.** We train a decoder to reconstruct the input image from the feature map $\tilde{Z}^{FC_i}$. The decoder is an enhanced version of the decoder used in [38]. We use the $L_1$ reconstruction loss for training and we train it on a small subset of the ImageNet-1k dataset, containing 20 randomly chosen classes. We evaluate the reconstruction quality by measuring the difference of the reconstructed and the input image using the $L_1$, $L_2$, peak signal to noise ratio (PSNR) and structural similarity index measure (SSIM) metrics.

**Results.** From Figure 4 we conclude that it is harder to reconstruct the input from feature maps $\tilde{Z}^{FC_i}$ at depth $i$, when using stochastic layers. The gap to the determinis-
Figure 6. Privacy-preserving federated learning. Stochastic layers offer increased privacy in at a cost of final task performance. A trade-off point must be chosen in accordance to the use-case.

remain private) to the centralized server, in order to learn a new task. We train a small classifier for each model on the CIFAR10/100 [49] datasets. The classifier contains 2 fully connected layers, followed by a softmax layer.

Results. In Figure 6 we visualize the trade-off between the accuracy on the new tasks and the input reconstruction. We observe that using stochastic layers offers better privacy in this setting at a cost of task performance. With this knowledge one can choose a trade-off point, in accordance with the final task and use-case.

5.5. Transfer Learning

Transfer learning is concerned with the problem of leveraging information from a rich source domain (with large labeled datasets), in order to improve the learning on a different downstream task, for which only limited data is available [19, 71]. For vision problems, the ImageNet [73]-trained models are established starting points for the transfer learning on natural images, due to the large variability and scale of the dataset.

Setup. We compare transferability of our features on a standard benchmark suite containing the aircraft [58], birdsnap [5], CIFAR10/100 [49], Caltech-101 [23], Caltech-256 [29], cars [48], DTD [11], flowers [63], food-101 [7], pets [66], and SUN-397 [93] datasets. We follow the setup of [74]. We use the AdamW optimizer [56] with an initial learning rate of $10^{-3}$ and weight decay of $5 \times 10^{-2}$. The transfer learning lasts for 150 epochs and the learning rate gets decayed by a factor of 10 every 50 epochs.

Results. Our findings are summarized in Table 3. Despite the tendency of big transformer architectures to quickly overfit, all compared methods obtain competitive results. We observe that our stochastic networks are able to retain their performance and sometimes even exceed the deterministic baseline. The results are quite robust to the level of noise applied. We therefore conclude that the transferability is not negatively impacted by introducing the stochasticity.

### Table 3. Transfer learning. The transfer learning capabilities of vision transformers are not negatively impacted by our stochastic layers. We even observe slight improvements for some datasets.

| Dataset     | Regular pre-training | $\Delta$ during pre-training |
|-------------|-----------------------|------------------------------|
|             |                       | 0.1 | 0.5 | 1.0 |
| Aircraft    | 72.34%                | 72.52% | 71.50% | 72.40% |
| Birdsnap    | 70.41%                | 69.97% | 69.58% | 70.00% |
| CIFAR-10    | 98.11%                | 98.36% | 98.13% | 98.29% |
| CIFAR-100   | 87.41%                | 87.93% | 87.65% | 88.04% |
| Caltech-101 | 91.89%                | 91.55% | 91.89% | 91.85% |
| Caltech-256 | 83.85%                | 83.87% | 84.07% | 83.99% |
| Cars        | 80.76%                | 82.13% | 81.52% | 81.99% |
| DTD         | 73.46%                | 73.72% | 73.94% | 73.94% |
| Flowers     | 92.39%                | 94.02% | 93.69% | 93.61% |
| Food        | 87.06%                | 87.45% | 87.10% | 87.29% |
| Pets        | 92.83%                | 93.10% | 93.16% | 93.08% |
| SUN397      | 62.34%                | 62.35% | 62.79% | 62.36% |

Wins: – | 10/12 | 10/12 | 10/12

6. Conclusion

In this work, we investigate the role of stochasticity in vision transformers. By interjecting linear stochastic layers during both training and inference, we transform the feature activations of each multilayer perceptron, while preserving the topological structure of the activations. The introduced stochasticity encourages the network to rely on topological features, that in turn offer the robustness and privacy of visual features, while retaining the original predictive performance. We demonstrated the utility of our features for the applications of adversarial robustness, network calibration, feature privacy, as well as for federated and transfer learning. From our experiments we conclude that our stochastic layers are well behaved and our visual features offer exciting results on those tasks.

Ethical and Societal Impact. This work is concerned with the learning of improved visual features by noise injection. Even though these features offer better resilience on privacy and adversarial attacks, they do not meet a certifiable standard as of now. Therefore they should not be used in a production system without additional layers of safety measures, such as encryption or humans-in-the-loop. Although our method allows for an improved estimation of prediction confidence overall, we did not investigate the behaviour with regards to specific subgroups of classes or people. This needs to be carefully analysed with appropriate datasets to mitigate discriminatory actions by a decision-making system based on our algorithm.
Stochastic Layers in Vision Transformers
Supplementary material

Nikola Popovic\textsuperscript{1}, Danda Pani Paudel\textsuperscript{1}, Thomas Probst\textsuperscript{1}, Luc Van Gool\textsuperscript{1,2}
\textsuperscript{1}Computer Vision Laboratory, ETH Zurich, Switzerland
\textsuperscript{2}VISICS, ESAT/PSI, KU Leuven, Belgium
\{nipopovic, paudel, probstt, vangool\}@vision.ee.ethz.ch

In this supplementary document, we first want to give additional details on our implementation and experimental setup. In the second part, we complete the results from the main paper, and also show additional comparisons and studies regarding the training process, privacy preservation, adversarial robustness, confidence calibration, and transfer learning. We round off the document by a discussion of the effect and rationale of our design choices, as well as several aspects of our experimental findings.

A. Additional implementation details

Training our stochastic layers When fine-tuning our network to adapt to the stochastic layers with \( \alpha^j \sim U(1 - \Delta, 1 + \Delta) \) Federated learning When performing the experiments mentioned in (from Equation (8) of the main paper), we gradually adjust \( \Delta \) from 0 to its final value during the first third of the training epochs. When we use Dropout as a baseline, while matching the variance of \( \alpha^j \sim U(1 - \Delta, 1 + \Delta) \), we also increase the drop probability from 0 to its final value in this manner. The same goes for the baseline which independently draws \( \alpha^j \sim U(1 - \Delta, 1 + \Delta) \) for each token – uniform everywhere. Also, in the experiment with adversarial training (see Section 5.3 of the main paper) and in experiments when we train from scratch (see Table 5), we use the same \( \Delta \) scheduling rule.

Privacy preservation When performing the experiments mentioned in, we train the decoder for 100 epochs with a batch size of 32. The encoder is the part of the transformer which produces the feature map and its weights are frozen. The stochasticity remains turned on during both training and inference.

Federated learning When performing the experiments mentioned in Section 5.4.2 of the main paper, the feature extractor is the part of the transformer which produces the feature map and its weights are frozen. The small classifier with 2 fully connected layers, followed by a softmax, is trained for 100 epochs, with the learning rate of 0.0001 and the AdamW optimizer. The learning rate is divided by 10 after 50 epochs. The stochasticity remains turned on during both training and inference.

B. Further analysis

In this section, we perform a further analysis of the experiments presented in Section 5 of the main paper.

In Table 4, we see classification accuracy on ImageNet-1k when using our stochastic layers, but also when using regular dropout and the uniform everywhere baseline. The training procedure is explained in Section 5.1 of the main paper. We conclude that there is no significant drop in accuracy when using our stochastic layers.

| \( \Delta \) | \( \Delta = 0 \) | \( N = 1 \) | \( N = 50 \) |
|----------|------------|------|------|
| Regular  | 79.61%     | 79.61% | 79.61% |
| Ours \( \Delta = 0.05 \) | 79.79% | 79.71% | 79.81% |
| Ours \( \Delta = 0.1 \) | 79.83% | 79.58% | 79.86% |
| Ours \( \Delta = 0.25 \) | 79.87% | 79.28% | 79.88% |

| \( \Delta \) | \( \Delta = 0.5 \) | Training | \( \Delta = 1.0 \) | \( \Delta = 0.75 \) |
|----------|----------------|--------|------------|------|
| Ours     | 79.60% | Uniform everywhere | 79.63% | 79.72% |
| Dropout  | 79.60% | Dropout | 75.54% | 79.79% |
| Ours \( \Delta = 0.75 \) | 78.93% | Uniform everywhere | 75.46% | 78.83% |

In Table 4, we see classification accuracy of networks which have been trained and evaluated on 100 randomly chosen classes of ImageNet-1k. The network has been trained from scratch for 300 epochs, following the settings in from [85]. For a brief discussion on this experiment, please look at Section C.1.

In Table 6, we see classification accuracy of the state-
of-the-art Swin transformers [55] on the ImageNet-1k. The training procedure is explained in Section 5.1 of the main paper. We observe that our stochastic layers also perform well with this architecture.

Now, we further analyse the problem discussed in Section 5.2 of the main paper. In Table 7, we notice that confidence calibration is much worse without temperature tuning. In Table 8 we see results obtained with temperature tuning. We observe that our stochastic layers help improve confidence calibration when used together with temperature tuning. We also observe that it achieve a better calibration than the regular dropout baseline and the uniform everywhere baseline. We also observe that the calibration is better for stronger noise, and when using monte-carlo sampling.

Next, we further analyse the problem discussed in Section 5.3 of the main paper. When dealing with stochasticity inside the network, Athalye et al. [2] demonstrated that for the class of randomness-based defenses, an expectation-over-transformation (EOT) attack is more effective. This can be viewed as using PGD (Equation (16) from the main paper) with the proxy gradient,

$$\mathbb{E}_{q(w)} \left[ \nabla_x \mathcal{L}(h(\hat{x}^{k-1}, w), l) \right] \approx \frac{1}{T} \sum_{t=1}^{T} \nabla_x \mathcal{L}(h(\hat{x}^{k-1}, w_t), l),$$

where $q(w)$ represents the distribution of the noise $w \sim q(w)$ injected into the randomized classifier $h(x, w)$.
Table 10. Adversarial robustness, with adversarial training. Our stochastic layers increase adversarial robustness in vision transformers. Combining the effects of strong noise with monte-carlo sampling can improve adversarial robustness and clean accuracy under adversarial training.

| Block        | Adversarial training with \(\epsilon = 2\) | Adversarial training with \(\epsilon = 4\) |
|--------------|--------------------------------|--------------------------------|
| Clean samples | PDG-10 attack | Clean samples | PDG-10 attack |
| Regular      | 71.64% | 24.84% | 64.04% | 27.24% |
| Ours, \(\Delta = 0.5\) | 71.75% | 44.31% | 66.83% | 30.08% |
| Ours, \(\Delta = 50\) | 73.74% | 46.21% | 68.05% | 31.47% |
| Uniform everywhere, \(\Delta = 50\) | 74.06% | 45.89% | 69.74% | 32.41% |
| Dropout \(\Delta = 1\) | 73.39% | 45.77% | 68.87% | 32.02% |
| Dropout \(\Delta = 50\) | 74.44% | 46.44% | 70.33% | 32.44% |

Table 11. Learning a classifier on CIFAR-10 with features before and after the activation function. Using features before the activation function to learn a classifier achieves better performance than when using features after the activation.

In Table 9 we can see the adversarial robustness against the EOT attack with 5 gradient samples within a 10 step PGD with 5 random restarts. For a brief discussion about these results, please look at Section C.6. In Table 10 we can see an extended version of Table 2a of the main paper.

In the following, we further analyse the problem discussed in Section 5.4.1 of the main paper. In Figure 7 we see examples of image reconstruction from the feature map of block 7 of the transformer. For a brief discussion of these results, please look at Section C.3.

We will now further analyse the problem discussed in Section 5.4.2 of the main paper. In Table 11 we observe the results of learning a classifier on CIFAR-10 with features before and after the activation function. We observe that using features before the activation function to learn a classifier achieves better performance than when using features after the activation. This justifies the choice of not using the features after the ReLU activation in federated learning experiments, even though they can be argued to be more private.

Finally, we further analyse the problem from Section 5.5 of the main paper. In Table 12 we observe that the transfer learning capabilities are not negatively impacted by our stochastic layers. We even observe slight improvements for some datasets.

Table 12. Transfer learning. The transfer learning capabilities of vision transformers are not negatively impacted by our stochastic layers. We even observe slight improvements for some datasets.

In Figures 8a, 8b and 8c we see histograms of barcode distances [10, 33] of feature maps before and after the stochasticity is injected. The feature maps are from the last block of the transformer. Histograms are constructed from 1000 randomly chosen input images from the validation set. The stochasticity level is \(\Delta = 0.5\). In Figures 8d, 8e and 8f we can see the barcode persistence diagrams for a randomly chosen input image for dropout, uniform everywhere and our method, respectively. For a brief discussion of this experiment, please see Section C.4.

In Figure 9 we see histograms of average barcode distances between 4 samples for our stochastic network and the deterministic baseline. Stochasticity with the level of \(\Delta = 0.5\) is turned on for both the stochastic network and the deterministic baseline as well. We take the feature map from the last block of the transformer, right before the stochasticity is injected. Histograms are constructed from 1000 randomly chosen input images from the validation set. For a brief discussion of this experiment, please see Section C.4.

C. Discussion

C.1. Difference to standard Dropout

**Theoretical** In its original formulation, Dropout [82] aims to prevent co-adaptation of neurons through randomly dropping activations, thus creating randomized neural pathways in fully-connected layers. In the case of CNNs, 2D dropout (SpatialDropout) [84] has found to be a more effective regularization, due to the high correlation of nearby pixels. In the case of vision transformers [20] however, information of neighbouring pixels is collapsed into token features (pixels within the same input patch) as well as across tokens (pixels of different input patches). Therefore, the concept of neighbourhood gets “diluted”, and the conclusions of [82] and [84] are in a state of conflict when it comes to vision transformers. On the other hand, our Stochastic Layers (LS) are designed for transformer-based architectures, and we draw our theoretical motivations from the homeomorphic nature
Figure 7. **Privacy preservation.** Reconstructed images from block 7 of the transformer with and without our stochastic layers. First row depicts the original image. The second row depicts images reconstructed from the feature map of a deterministic network. Finally, the third row depicts reconstructions from our stochastic network. Injecting noise preserves more privacy, in particular in the facial details.

Figure 8. **Persistence barcode diagrams before and after injecting stochasticity.** We see histograms of barcode distances of feature maps before and after the stochasticity is injected. The histograms in Figures 8a, 8b and 8c correspond to using simplex-0, simplex-1 and simplex-2, respectively. The feature maps are from the last block of the transformer. Histograms are constructed from 1000 randomly chosen input images from the validation set. The stochasticity level is $\Delta = 0.5$. In Figures 8d, 8e and 8f we can see the barcode persistence diagrams for a randomly chosen input image for dropout, uniform everywhere and our method, respectively. Based on small distances of our method before and after injecting stochasticity, we conclude that the topological structure of token features, like it was mentioned in Section 3 of the main paper.
stochasticity draws for the same input. We see histograms of average barcode distances between 4 stochasticity draws for the same input image. Stochasticity with the level of $\Delta = 0.5$ is turned on for both the stochastic network and the deterministic baseline as well. The histograms in Figures 8a, 8b and 8c correspond to using simplex-0, simplex-1 and simplex-2, respectively. We take the feature map from the last block of the transformer, right before the stochasticity is injected. Histograms are constructed from 1000 randomly chosen input images from the validation set. We observe that feature maps of our network have a more consistent topology (lower barcode distances) compared to the deterministic baseline, when stochasticity is injected.

of their operation – by applying the same noise on all tokens (see Section 3 of the main paper). Furthermore, to obtain stochastic homeomorphisms, we avoid noise distributions that violate invertibility of the operation, in particular the Bernoulli noise employed by [82, 84]. We therefore choose uniform noise in our experiments.

Experimental In practice, we observe the different effects of our LS and Dropout in several settings. For instance, our LS method offers better confidence calibration than Dropout and uniform everywhere, as shown in Table 8. This pattern coincides with several observations on other tasks. For example, in Figure 5 of the main paper, it can be observed that injecting stochasticity through non-homeomorphic operations preserves less privacy. Also, our LS provides stronger stochasticity to the network under the same variance, even though the random variables are the same for each token. This can be observed on ImageNet-1k classification accuracy shown in Table 4. The non-homeomorphic operations Dropout and uniform everywhere tend to retain more accuracy of the original network (e.g. with $N = 1$ sample).

C.2. Fine-tuning vs. training from scratch

Fine-tuning brings the benefit of being able to quickly integrate our stochastic layers into an already trained network. In this process, the architecture of the original transformer is unchanged, while at the same time we can enjoy the benefits discussed in Section 5 of the main paper. This saves a lot of training cost and CO2 emissions. This does not significantly degrade ImageNet-1k classification accuracy as can be seen in Figure 1a of the main paper. Moreover, to demonstrate that training the network from scratch with our stochastic layers is possible, even with strong stochasticity, we have performed the experiments shown in Table 5. We observe that our stochastic layers also behave well in this setting. In these experiments, the networks have been trained from scratch for 300 epochs on 100 randomly chosen classes of the ImageNet-1k dataset.

C.3. Privacy

In Figure 7, we show that injecting noise preserves more privacy. Reconstructed images from block 7 of the transformer with and without our stochastic layers are shown. The first row depicts the original image. The second row depicts images reconstructed from the feature map of a deterministic network. Finally, the third row depicts reconstructions from our stochastic network. In particular the facial details are more degraded by our stochastic layers.

C.4. Topology

In Figure 8 we show histograms of barcode distances of feature maps before and after the stochasticity is injected. Based on small distances of our method before and after injecting stochasticity, we conclude that they the topological structure of token features, like it was mentioned in Section 3 of the main paper.

In Figure 9 We see histograms of average barcode distances between 4 stochasticity draws for the same input image for our stochastic network and the deterministic baseline. We observe that feature maps of our network have a more consistent topology (lower barcode distances) compared to the deterministic baseline, when stochasticity is injected. Therefore, we conclude that the topological structure is

C.5. Regularization effect

It is known that stochasticity can have a regularizing effect [82]. In fact, our results in Table 5 hint in this direction when training from scratch. However, we did not further investigate this behaviour due to the reasons discussed in Section C.2 and also because it is not the main focus of this paper.

C.6. Expectation over Transformation attack

In practice, the EOT attack comes with a very high cost, since for every step of the attack one needs to compute multiple complete forward and backward passes for the same input. Furthermore, since our stochastic layers may be easily introduced to most of the popular transformer networks, the attacker can know the architecture and the weights but would still be oblivious to the presence of noise and monte-carlo sampling during inference.

Bearing this in mind, we analyze the impact of the EOT attack with 5 gradient samples within a 10 step PGD with 5 random restarts. The results are presented in Table 9. Even
under this strong attack, which exploits the stochasticity, we are able to retain the robustness of the regular baseline.

C.7. Effect of transformer architecture

To demonstrate the generality of our stochastic layers, we also conducted experiments on the state-of-the-art Swin transformer [55]. The results can be seen in Table 5. Our stochastic layers are integrated in the exact same way as they are integrated into the DeiT-S vision transformer architecture. We observe that they also behave well with this architecture.

References

[1] Alexander Amir Alemi, Ian S. Fischer, Joshua V. Dillon, and Kevin P. Murphy. Deep variational information bottleneck. ArXiv, abs/1612.00410, 2017. 2
[2] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In ICML, 2018. 6, 10
[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. 3
[4] Samyadeep Basu, Rauf Izzmailov, and Chris Mesterharm. Membership model inversion attacks for deep networks. ArXiv, abs/1910.04257, 2019. 7
[5] Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L. Alexander, David W. Jacobs, and Peter N. Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. 2014 IEEE Conference on Computer Vision and Pattern Recognition, pages 2019–2026, 2014. 8
[6] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. arXiv preprint arXiv:1902.01046, 2019. 6
[7] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 - mining discriminative components with random forests. In ECCV, 2014. 8
[8] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot to resist adversarial examples. 2018. 6
[9] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In ECCV, pages 213–229. Springer, 2020. 2
[10] Frédéric Chazal and Bertrand Michel. An introduction to topological data analysis: fundamental and practical aspects for data scientists. Frontiers in Artificial Intelligence, 4, 2021. 2, 11
[11] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. 2014 IEEE Conference on Computer Vision and Pattern Recognition, pages 3606–3613, 2014. 8
[12] Moustapha Cissé, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. ArXiv, abs/1704.08847, 2017. 6
[13] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. In ICML, 2019. 2, 6
[14] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provably robustness of relu networks via maximization of linear regions. ArXiv, abs/1810.07481, 2019. 6
[15] Ekin Dogus Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. RandAugment: Practical automated data augmentation with a reduced search space. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 3008–3017, 2020. 1, 2
[16] Bin Dai, Chen Zhu, and David P. Wipf. Compressing neural networks using the variational information bottleneck. ArXiv, abs/1802.10399, 2018. 2
[17] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In Annual Meeting of the Association for Computational Linguistics (ACL), pages 2978–2988, 2019. 2
[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In NAACL-HLT, pages 4171–4186, 2019. 2
[19] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In ICML, 2014. 8
[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In ICLR, 2021. 1, 2, 3, 11
[21] Joshua J. Engelsma, Anil K. Jain, and Vishnu Naresh Bodeeti. Hers: Homomorphically encrypted representation search. ArXiv, abs/2003.12197, 2020. 7
[22] Hao Sen Andrew Fang and Quan Qian. Privacy preserving machine learning with homomorphic encryption and federated learning. Future Internet, 13:94, 2021. 7
[23] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In CVPR Workshops, 2004. 8
[24] Chris Finlay and Adam M. Oberman. Scaleable input gradient regularization for adversarial robustness. ArXiv, abs/1905.11468, 2019. 6
[25] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. ArXiv, abs/1506.02158, 2015. 1, 2, 5
[26] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. ArXiv, abs/1506.02142, 2016. 1, 2, 5
[27] Marcel Geppert, Viktor Larsson, Pablo Speciale, Johannes L. Schönberger, and Marc Pollefeys. Privacy preserving structure-from-motion. In ECCV, 2020. 6, 7
[28] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015. 5
[29] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007. 8
[30] Shixiang Shane Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *CoRR*, abs/1412.5068, 2015. 6
[31] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017. 4, 5
[32] Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018. 7
[33] William H Guss and Ruslan Salakhutdinov. On characterizing the capacity of neural networks using algebraic topology. *arXiv preprint arXiv:1802.04443*, 2018. 2, 11
[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 4
[35] Chao-Yung Hsu, Chun-Shien Lu, and Soo-Chang Pei. Image feature extraction in encrypted domain with privacy-preserving sift. *IEEE Transactions on Image Processing*, 21:4593–4607, 2012. 7
[36] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth, 2016. 4
[37] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksandar Madry. Adversarial examples are not bugs, they are features. *arXiv*, abs/1905.02175, 2019. 5
[38] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2018. 7
[39] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018. 1
[40] Daniel Jakubovitz and Raja Giryes. Improving dnn robustness to adversarial attacks using jacobian regularization. *ArXiv*, abs/1803.08680, 2018. 6
[41] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019. 6
[42] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. 2019 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405, 2019. 2
[43] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NIPS*, 2017. 1, 2, 5
[44] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *arXiv preprint arXiv:2101.01169*, 2021. 2
[45] Diederik P. Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *ArXiv*, abs/1506.02557, 2015. 2
[46] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014. 1, 2
[47] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *ArXiv*, abs/1610.05492, 2016. 6
[48] Jonathan Krause, Jun Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013. 8
[49] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 8
[50] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *ArXiv*, abs/1611.01236, 2017. 6
[51] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles, 2017. 5
[52] Max-Heinrich Laves, Sontje Ihler, Karl-Philipp Kortmann, and Tobias Ortmaier. Calibration of model uncertainty for dropout variational inference. *ArXiv*, abs/2006.11584, 2020. 2
[53] Sungyoon Lee, Hoki Kim, and Jaewook Lee. Gradddiv: Adversarial robustness of randomized neural networks via gradient diversity regularization. *ArXiv*, abs/2107.02425, 2021. 6
[54] Tom Ching LingChen, Ava Khonsari, Amireza Lashkari, Mina Rafi Nazari, Jaspreet Singh Sambee, and Mario A. Nascimento. Uniformaugment: A search-free probabilistic data augmentation approach. *ArXiv*, abs/2003.14348, 2020. 2
[55] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021. 10, 14
[56] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017. 4, 8
[57] Aleksandar Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ArXiv*, abs/1706.06083, 2018. 6
[58] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013. 8
[59] Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Ann Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. *ArXiv*, abs/2106.07998, 2021. 4, 5
[60] Gonçalo Mordido, Haojin Yang, and Christoph Meinel. Dropout-gan: Learning from a dynamic ensemble of discriminators. *ArXiv*, abs/1807.11346, 2018. 2
[61] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. *Proceedings of the ... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, 2015:2901–2907, 2015. 5
[62] Arvind Neelakantan, Luke Vilnis, Quoc V. Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks. ArXiv, abs/1511.06807, 2015. 2

[63] Maria-Elena Nilisback and Andrew Zisserman. Automated flower classification over a large number of classes. 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing, pages 722–729, 2008. 8

[64] Alexander Ororbia, C. Lee Giles, and Daniel Kifer. Unifying adversarial training algorithms with flexible deep data gradient regularization. ArXiv, abs/1601.07213, 2016. 6

[65] Konstantinos Panousis, Sotirios Chatzis, Antonios Alexos, and Sergios Theodoridis. Local competition and stochasticity for adversarial robustness in deep learning. In International Conference on Artificial Intelligence and Statistics, pages 3862–3870. PMLR, 2021. 1

[66] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. 2012 IEEE Conference on Computer Vision and Pattern Recognition, pages 3498–3505, 2012. 8

[67] Rafael Pinot, Laurent Meinier, Alexandre Araujo, Hisashi Kashima, Florian Yger, Cédrick Gouy-Pailler, and Jamal Atif. Theoretical evidence for adversarial robustness through randomization. In NeurIPS, 2019. 6

[68] F. Pittaluga, Sanjeev J. Koppal, Sing Bing Kang, and Sudipta N. Sinha. Revealing scenes by inverting structure from motion reconstructions. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 145–154, 2019. 7

[69] Zhan Qin, Jingbo Yan, Kui Ren, Chang Wen Chen, and Cong Wang. Towards efficient privacy-preserving image feature extraction in cloud computing. Proceedings of the 22nd ACM international conference on Multimedia, 2014. 7

[70] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 588–597, 2019. 1, 2, 6

[71] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 512–519, 2014. 8

[72] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In AAAI, 2018. 6

[73] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 115(3):211–252, 2015. 4, 8

[74] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better?. 2020. 8

[75] Hadi Salman, Greg Yang, Junghsian Li, Pengchuan Zhang, Huan Zhang, Ilya P. Razenshteyn, and Sébastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. In NeurIPS, 2019. 6

[76] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John P. Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In NeurIPS, 2019. 6

[77] Kumar Shridhar, Felix Laumann, and Marcus Liwicki. A comprehensive guide to bayesian convolutional neural network with variational inference. arXiv preprint arXiv:1901.02731, 2019. 1

[78] Abhishek Singh, Ayush Chopra, Vivek Sharma, Ethan Garza, Emily Zhang, Praneeth Vepakomma, and Ramesh Raskar. Disco: Dynamic and invariant sensitive channel obfuscation for deep neural networks. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 12120–12130, 2021. 7

[79] Samuel L. Smith, Benoit Richard Umbert Dherin, David G. T. Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. ArXiv, abs/2101.12176, 2021. 1, 2

[80] James C Spall. Introduction to stochastic search and optimization: estimation, simulation, and control, volume 65. John Wiley & Sons, 2005. 1

[81] Pablo Speciale, Johannes Schönberger, Sing Bing Kang, Sudipta N. Sinha, and Marc Pollefeys. Privacy preserving image-based localization. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 5488–5498, 2019. 6, 7

[82] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958, 2014. 1, 2, 4, 7, 11, 13

[83] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, D. Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. CoRR, abs/1312.6199, 2014. 5, 6

[84] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 648–656, 2015. 11, 13

[85] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In International Conference on Machine Learning, volume 139, pages 10347–10357, July 2021. 3, 4, 9

[86] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Deep image prior. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9446–9454, 2018. 7

[87] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vish-
Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.

[88] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. arXiv preprint arXiv:1812.00564, 2018.

[89] Pei Wang, Yijun Li, Krishna Kumar Singh, Jingwan Lu, and Nuno Vasconcelos. Imagine: Image synthesis by image-guided model inversion. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 3680–3689, 2021.

[90] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. arXiv preprint arXiv:2102.12122, 2021.

[91] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In CVPR, 2021.

[92] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. ArXiv, abs/2001.03994, 2020.

[93] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 3485–3492, 2010.

[94] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In NIPS, pages 5753–5763, 2019.

[95] Hongxu Yin, Pavlo Molchanov, Zhizhong Li, José Manuel Alvarez, Arun Mallya, Derek Hoiem, Niraj Kumar Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deep inversion. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 8712–8721, 2020.

[96] Tianyuan Yu, Yongxin Yang, Da Li, Timothy Hospedales, and Tao Xiang. Simple and effective stochastic neural networks. 2019.

[97] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Token-to-token ViT: Training vision transformers from scratch on imagenet. arXiv preprint arXiv:2101.11986, 2021.

[98] Runtian Zhai, Chen Dan, Di He, Huan Zhang, Boqiang Gong, Pradeep Ravikumar, Cho-Jui Hsieh, and Liwei Wang. Macer: Attack-free and scalable robust training via maximizing certified radius. ArXiv, abs/2001.02378, 2020.

[99] Hongyang R. Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In ICML, 2019.

[100] Yuheng Zhang, R. Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Xiaodong Song. The secret revealer: Generative model-inversion attacks against deep neural networks. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 250–258, 2020.

[101] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H.S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In CVPR, 2021.