Probabilistic Spatial Transformer Networks

Abstract

Spatial Transformer Networks (STNs) estimate image transformations that can improve downstream tasks by "zooming in" on relevant regions in an image. However, STNs are hard to train and sensitive to mis-predictions of transformations. To circumvent these limitations, we propose a probabilistic extension that estimates a stochastic transformation rather than a deterministic one. Marginalising transformations allows us to consider each image at multiple poses, which makes the localisation task easier and the training more robust. As an additional benefit, the stochastic transformations act as a localised, learned data augmentation which improves the downstream tasks. We show across standard imaging benchmarks and on a challenging real world dataset that these two properties lead to improved classification performance, robustness and model calibration. We further demonstrate that the approach generalises to non-visual domains by improving model performance on time-series data.

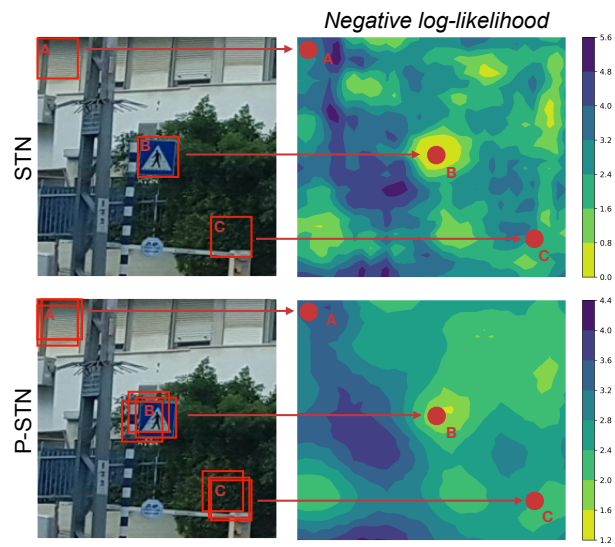

Figure 1: The Probabilistic Spatial Transformer Network (P-STN) marginalises over a distribution of possible input transformations. By "looking in multiple places" we hope to stabilise the brittle nature of the regular spatial transformer: The P-STN loss landscape is significantly more smooth and with fewer local minima compared to the STN.

1 INTRODUCTION

The *Spatial Transformer Network (STN)* [Jaderberg et al., 2015] predicts a *transformation* on input data in order to simplify a downstream task. For example, a neural network might benefit from e.g. "zooming in" on relevant parts of an image, remove unwarranted image rotations, or time normalise sequence data before making predictions. In principle, this allows improvements in both robustness, interpretability and efficiency of the model. In practice, the situation is, however, not as ideal. Both at training and test time, the STN is sensitive to small mis-predictions of transformations. For example, if the STN zooms in on the wrong part of an image, then the signal is lost for the downstream task, e.g. see crop A and C in Fig. 1. The empirical impact is that STNs are difficult to train and often do not live up to their promise.

From a probabilistic perspective, this sensitivity has an obvious solution: we should estimate the posterior of the applied transformation and marginalise accordingly. This amounts to 'trying many different transformations', and should improve robustness. It is exactly this approach we investigate.

STNs consist of two parts. A localisation network performs the transformation task, i.e. estimates the transformation parameters θ for a given image I and applies the corresponding transformation $T_\theta(I)$. A standard neural network performs the downstream task on the transformed image, i.e. computing $p(y|T_\theta(I))$. Since we are concerned with classification

Submitted to the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022). **To be used for reviewing only**.

tasks we will refer to the latter as the classifier, but note that the approach generalises to other tasks.

In our probabilistic STN (P-STN), we estimate a distribution over transformations that we marginalise: $p(y|I) = \int p(y|T_\theta(I))\mathrm{d}\theta$. We approximate this intractable integral via Monte Carlo, i.e. we sample transformations. Those transformation samples produce different transformed versions of the input image, $\{T^i_\theta(I)\}_{s=1\ldots S}$. The classifier makes predictions on all samples, and we aggregate the predictions. Figure 2 shows the model architecture.

We hypothesise that marginalising image transformation has benefits for both parts of the model. For the *localisation* network, our model gets to 'try many different transformations' through random sampling. This should improve the localisation. Secondly, the classifier now gets presented with different transformed versions of the input image through Monte Carlo samples $\{T^i_\theta(I)\}_{s=1\ldots S}$. Interestingly, this corresponds to a type of data augmentation, which should improve classification.

We verify the hypothesis by making the following contributions:

1. We develop the Probabilistic Spatial Transformer; a hierarchical Bayesian model over image transformations.
2. Perform variational inference to fit the transformation model as well as downstream model end-to-end, using only label information.
3. Experimentally demonstrate that our model achieves better localisation, increased classification accuracy (resulting from learned data augmentation) and improved calibration.

2 RELATED WORK

Spatial transformer networks apply a spatial transformation to the input data as part of an end-to-end trained neural network [Jaderberg et al., 2015]. The transformation parameters are estimated from each input separately through a neural network. Most commonly, STNs implement simple affine transformations, such that the network can learn to zoom in on relevant parts of an image before solving the task at hand. STNs have shown themselves to be useful for both generative and discriminative tasks, using both affine and more complex transformations, and have seen uses in different data modalities [Jaderberg et al., 2015, Detlefsen and Hauberg, 2019, Detlefsen et al., 2018, Shapira Weber et al., 2019, Sønderby et al., 2015, Lin and Lucey, 2016, Kanazawa et al., 2016]. We propose a probabilistic extension of this idea, replacing the usual likelihood maximization with marginalisation over transformations.

Bayesian Deep Learning aims to solve probabilistic computations in deep neural networks. Priors are put on weights and marginalised at training and test times, often yielding useful uncertainties in the posterior predictive. This is in general intractable, and approaches differ mainly in the type of approximation to the weight posterior. Gal and Ghahramani [2016] propose to view dropout as a Bernoulli approximation to the weight posterior (i.e. randomly switching each weight on or off). The Laplace approximation [MacKay, 1992, Daxberger et al., 2021] places a Gaussian posterior over a trained neural network's weights. Another generally successful way to obtaining predictive uncertainties is to simply train an ensemble of models. Originally proposed as an alternative to Bayesian DL [Lakshminarayanan et al., 2017], the approach can be interpreted in the Bayesian framework by interpreting the weights of the trained ensemble members as samples from a weight posterior [Gustafsson et al., 2020]. Similar to our method, Blundell et al. [2015] choose a variational approach with a simple Gaussian, mean field posterior over weights. Our approach, differs from standard Bayesian DL in that we are not reasoning about distributions over neural network weights $p(w)$, but instead a subnetwork's (i.e. the localiser's) *outputs* $p(\theta)$. Drawing from the posterior over image transformations, we effectively recover data augmentation.

Data augmentation (DA) is an useful way to increase the amount of available data [LeCun et al., 1995, Krizhevsky et al., 2012]. DA requires prior knowledge about the structure of the data: the target y_n is invariant to certain transformations of the observation x_n. Invariance assumptions are usually straight forward for natural images, thus DA is particularly common for image data, where the transformation family is most commonly chosen to be rotations, scalings, and similar [Goodfellow et al., 2009, Baird, 1992, Simard et al., 2003, Krizhevsky et al., 2012, Loosli et al., 2007]. The general trend is that, beyond 'intuitive' data, such as images, gathering an invariance prior is difficult, and DA is often hard to realise through manual tuning.

Learned data augmentation provides a more principled approach to artificially extending datasets. Hauberg et al. [2016] estimate an augmentation scheme from the training data via pre-aligning images in an unsupervised manner. The approach allows for significantly more complex transformations than the usual affine family, but the unsupervised nature, and thus the two step training process, renders the approach suboptimal. Similarly, Cubuk et al. [2019, 2020] use reinforcement learning and grid search to learn data augmentation schemes, but rely on on validation data rather than an end-to-end formulation.

Learning data augmentation end-to-end requires a loss function suitable for model selection as we are effectively trying to learn an inductive bias. Based on this realisation, Van der Wilk et al. [2018] learn DA end-to-end in Gaussian processes (GPs) via the marginal likelihood, a suitable loss for model selection and thus invariance learning [MacKay, 2003]. The marginal likelihood is hard to compute for NNs,

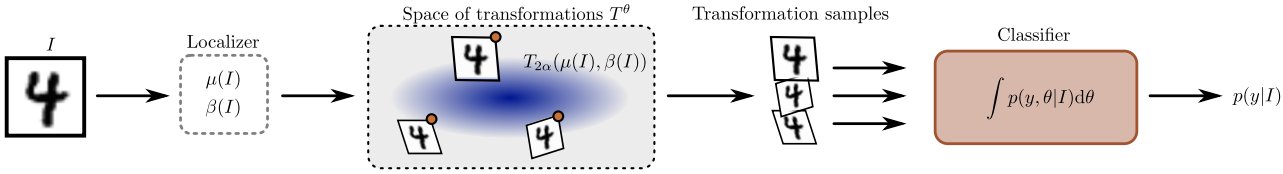

Figure 2: The P-STN pipeline. From the observed image I, a distribution of transformations is estimated. Samples from this distribution are applied to the observed image to produce augmented samples, which are fed to a classifer that averages across samples. In the deterministic STN case, the localiser only computes one transformation $\theta(I)$, which can be thought of as the maximum likelihood solution. Instead of the multiple transformation samples, we obtain a single $T_\theta(I)$ in this case.

so Schwöbel et al. [2021] extend this idea to NNs by considering a Deep Kernel model, i.e. a neural network with a GP in the last layer. Benton et al. [2020] instead use the standard, maximum likelihood loss and explicitly regularise towards non-zero augmentations. Our model differs from existing data augmentation approaches – learned and non-learned – in that we estimate local, i.e. *per-image* transformations instead of a global augmentation scheme.

3 BACKGROUND

The STN localiser module estimates a transformation $\theta(I)$ that transforms a coordinate grid and interpolates an image accordingly. The classifier module takes the transformed image and computes $p(y|T_\theta(I))$. Both the localiser and classifier are neural networks. The STN can be trained end-to-end with only label information as long as the image transformations are parametrised in a differentiable manner.

Affine Transformations are a simple class of transformations that can be differentiably parametrised. We limit ourselves to the subset of affine transformations containing rotation, isotropic scaling and translation in x and y. In two dimensions (and the corresponding three-dimensional homogeneous coordinates), we thus learn $\theta = (r, s, t_x, t_y)$ which parametrises the affine matrix

$$A_\theta = \begin{bmatrix} s \cdot \cos r & -s \cdot \sin r & t_x \\ s \cdot \sin r & s \cdot \cos r & t_y \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3},\ s > 0. \quad (1)$$

Since $\det(A_\theta) = s^2$, the constraint $s > 0$ ensures invertibility and can be implemented as seen in Detlefsen et al. [2018]. In practice, the STN estimates such well-behaved, non-collapsing transformations without implementing the constraint explicitly. $T_\theta(I_{\text{obs}})$ is applied by transforming a grid of the target image size by A_θ and interpolating the source image at the resulting coordinates (see Jaderberg et al. [2015] for details).

Diffeomorphic Transformations (i.e. transformations that are differentiable, invertible and possess a differentiable inverse) are more general than affine transformations, and are not limited to the spatial domain. Freifeld et al. [2017] construct diffeomorphisms from continuous piecewise-affine velocity fields as follows. The transformation domain Ω is divided into subsets and an affine matrix is defined on each cell c of such a tesselation. Each affine matrix A_{θ_c} induces a vector field mapping each point $x \in c$ to a new position $v^{\theta_c} : x \mapsto A_{\theta_c} x$. These velocity fields are then integrated to form a trajectory for each image point x

$$\phi^\theta(x; 1) = x + \int_0^1 v^\theta(\phi(x; \tau))\mathrm{d}\tau.$$

Given boundary and invertibility constraints [Freifeld et al., 2017], such a collection of affine matrices $\{A_{\theta_c}\}_{c\subset\Omega}$ defines a diffeomorphic transformation $T^\theta : x \mapsto \phi^\theta(x, 1)$.

The libcpab library [Detlefsen, 2018] provides an efficient implementation for this approach, specifically optimised for use in a deep learning context where fast gradient evaluations are crucial. The author successfully employs CPAB-transformations within a Spatial Transformer Network [Detlefsen et al., 2018].

4 PROBABILISTIC SPATIAL TRANSFORMER

The P-STN is a probabilistic extension of the STN, where we replace the deterministic transformation $\theta(I)$ with a posterior over transformations $p(\theta|I)$. Figure 2 illustrates the proposed pipeline. We assume observed data of the form $\mathcal{D} = \{y_i, I_i\}_{i=1}^N$, where y is the target variable (e.g. class label), and I are observations of the covariates. For presentation purposes, we will consider the latter to be images, but the approach applies to any spatio-temporal data.

4.1 THE MODEL

Recall that STNs are trained end-to-end for the downstream task using only label information. Thus, while we observe y, θ is a latent variable. We model it to be governed by a second latent variable λ. λ is a precision parameter, effectively stopping the localisation distribution (i.e. the amount of 'data augmentation' we introduce) from collapsing. The necessity for non-collapsing augmentation is meticulously discussed in the literature [Benton et al., 2020, Van der Wilk et al., 2018, Schwöbel et al., 2021].

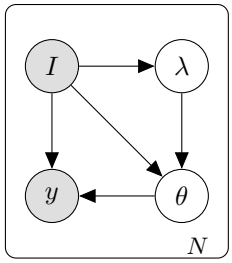

Figure 3: A graphical representation of the model structure. Grey nodes are observables and white are latents.

We wish to infer the latent variables in a Bayesian manner. This entails computing the (log-)marginal likelihood of the observed

$$\log p(I, y) = \log \iint p(I, y, \theta, \lambda)\, \mathrm{d}\theta\, \mathrm{d}\lambda. \tag{2}$$

We let the joint distribution factorise as (see Fig. 3)

$$p(y, I, \theta, \lambda) = p(y|I, \theta, \lambda)p(I, \theta, \lambda) \tag{3}$$
$$= p(y|I, \theta)p(\theta|\lambda, I)p(\lambda)p(I). \tag{4}$$

Notice $p(I)$ is unaffected by model parameters λ and θ, and in this sense can be specified without affecting the model. We define the prior over (θ, λ) semi-empirical as the prior over θ depends on observed covariates in the following way

$$p(\theta|\lambda, I) = \mathcal{N}(\theta|\mu(I), 1/\lambda), \tag{5}$$

where $\mu(I)$ is a function parametrised by a neural network, i.e. $\mu(I) := \mu_\Phi(I)$ for model parameters Φ. The prior over λ is a Gamma distribtion, i.e.

$$p(\lambda_i) = \Gamma(\alpha_0, \beta_0). \tag{6}$$

We remark here that there is one λ_i associated to each observation, and they are assumed to factorise: $p(\lambda) = \prod_{i=1}^N p(\lambda_i)$. This choice of conjugate priors for variance estimation is similar to [Stirn and Knowles, 2020, Takahashi et al., 2018, Detlefsen et al., 2019]. Finally, we assume that, conditional on I and θ, we have marginal independence in y, i.e. $p(y|I, \theta) = \prod_{i=1}^N p(y_i|I_i, \theta_i)$.

4.2 VARIATIONAL APPROXIMATION

The integral equation (2) for the marginal likelihood is intractable and, thus, the posterior $p(\lambda, \theta|I, y)$ is too. We derive a lower bound on the log marginal likelihood to utilise variational inference [Blei et al., 2017]. We choose the variational approximation q of the posterior $p(\theta, \lambda|I, y)$ as

$$q(\theta, \lambda) := p(\theta|\lambda, I)q(\lambda). \tag{7}$$

Here $p(\theta|\lambda, I)$ is given as before and $q(\lambda) := \prod_{i=1}^N \Gamma(\alpha_i, \beta(I_i))$. In our approximation, β is a neural network: hence, we use amortised inference in a similar way to the VAE model [Kingma and Welling, 2014].

We derive our lower bound using Jensen's inequality

$$\log p(y, I) = \log \iint p(y, I, \theta, \lambda)\mathrm{d}\theta\mathrm{d}\lambda \tag{8}$$
$$\geq \iint \log\left(\frac{p(y, I, \theta, \lambda)}{q(\theta, \lambda)}\right) q(\theta, \lambda)\mathrm{d}\theta\mathrm{d}\lambda \tag{9}$$
$$= \iint \log\left(\frac{p(y|I, \theta)p(\lambda)p(I)}{q(\lambda)}\right) p(\theta|\lambda, I)q(\lambda)\mathrm{d}\theta\mathrm{d}\lambda$$
$$= \underbrace{\mathbb{E}_{q(\theta,\lambda)} \log p(y|I, \theta)}_{\textbf{classification loss}} + \log p(I) - \mathrm{KL}(q(\lambda)\|p(\lambda))\,. \tag{10}$$

Thus, our evidence lower bound (ELBO) objective function (10), consists of two terms: a classification loss and a KL-term controlling the distance of the approximate posterior to the prior. During inference we can disregard $\log p(I)$ as it does not depend on parameters of interest.

4.3 INFERENCE

The choice of variational posterior implies the following for the **classification loss**

$$\mathbb{E}_{q(\theta,\lambda)} \log p(y|I, \theta) \tag{11}$$
$$= \iint \log p(y|I, \theta)q(\theta, \lambda)\mathrm{d}\theta\mathrm{d}\lambda \tag{12}$$
$$= \iint \log p(y|I, \theta)p(\theta|\lambda, I)q(\lambda)\mathrm{d}\theta\mathrm{d}\lambda \tag{13}$$
$$= \int \log p(y|I, \theta) \int \mathcal{N}(\theta|\mu(I), \lambda)\Gamma(\lambda|\alpha, \beta(I))\mathrm{d}\lambda\, \mathrm{d}\theta$$
$$= \int \log p(y|I, \theta) t_{2\alpha}(\theta|\mu(I)), \tfrac{\beta(I)}{\alpha})\mathrm{d}\theta. \tag{14}$$

Here t denotes a scaled and location-shifted Student's t-distribution with mean $\mu(I)$, scaling β, and α degrees of freedom. For clarity, the marginalised $q(\theta)$ is t-distributed. Here $p(y|I, \theta)$ is what previously was referred to as $p(y|T_\theta(I))$, i.e. the classifier conditioned the transformed I.

We approximate Eq. 14 using an unbiased estimate

$$\mathbb{E}_{q(\theta,\lambda)} \log p(y_i|I_i, \theta_i) \approx \frac{1}{S}\sum_{s=1}^S \log p(y_i|I_i, \theta_{i,s}), \tag{15}$$
$$\text{with } \theta_{i,s} \sim t_{2\alpha_i}(\cdot|\mu(I_i), \alpha_i, \beta(I_i)) \tag{16}$$

and backpropagate through neural networks $\mu(I)$ and $\beta(I)$ with the reparametrisation trick. In all experiments $\alpha_i = 1$.

Combining terms, the final ELBO we maximise becomes

$$\mathcal{L}_{p,q}(I, y) \approx \sum_{i=1}^N \frac{1}{S}\sum_{s=1}^S \log p(y_i|I_i, \theta_{i,s}) - \mathrm{KL}\left(q(\lambda)||p(\lambda)\right) + \text{const}, \tag{17}$$

which is readily optimised using any gradient-based method. The KL-term is analytically tractable and differentiable between two gamma distributions.

In practice, following Higgins et al. [2016] we introduce a weight parameter w to the KL-term. This requires us to tune w but in turn makes the model robust to the choice of prior. We perform a grid-search on a validation set to find the optimal w. Alternative to this, we could have done a grid search over β_0; instead we choose $\alpha_0 = \beta_0 = 1$ for all experiments. Similar to Kingma and Welling [2014], we find it often sufficient to draw only $S = 1$ samples during training. Note that our model naturally implies marginalisation, and correspondingly data augmentation, at *test-time* as well as the usual training time. At test time, we draw $S = 10$ transformation samples.

5 EXPERIMENTS & RESULTS

Our model consists of two parts, the classifier $p(y|T_\theta(I))$ and the probabilistic localiser that estimates the distribution over transformations. In the following experiments, we attempt to disentangle our model's benefits on the localisation (Sec. 5.1), the classification tasks (Sec. 5.2) and the model's calibration (Sec. 5.3).

The probabilistic localiser estimates $q(\theta) = t_2(\theta|\mu(I), \beta(I))$, i.e. in practice we implement a mean and a variance network, $\mu(I)$ and $\beta(I)$, respectively (see Fig. 2 for the architecture). We employ a small convolutional network (`Conv2d`, `Maxpool2d`, `ReLU`, `Conv2d`, `Maxpool2d`, `ReLU`) followed by two fully connected layers for both the localiser and classifier unless stated otherwise. The P-STN localiser has two heads. The number of parameters is stated in each experimental subsection. Unless stated otherwise, we keep the amount of parameters constant, i.e. when adding a localisation network we remove the extra parameters from the classifier for fair comparison.

Our model is implemented in PyTorch and experiments are run on 12 GB Nvidia Titan X GPUs. Code will be made available on github.

5.1 MARGINALISING TRANSFORMATIONS IMPROVES LOCALISATION ACCURACY

The appeal of STN models is that they are trained end-to-end, i.e. based only on labels for the downstream task, and not the transformations themselves. This same property, however, is what makes the STN hard to fit. The only signal we obtain is through the supervised downstream task (i.e. the classification labels) and thus gradient information is sparse. We will now investigate whether estimating a posterior over transformations and marginalising, i.e. 'getting to try multiple transformations', make the task easier.

In order to disentangle the localisation from the classification task we construct the following experiments. We first train a CNN on a pose-normalised dataset (regular MNIST and Fashion MNIST). We then generate a new dataset by randomly sampling transformations θ_{true} and applying them to the MNIST images. By saving the transformations used to generate the datasets, we now have transformation ground truth available. We freeze the CNN weights and train STN and P-STN with this fixed classifier, effectively learning to recover and 'undo' the true transformations.

5.1.1 Rotated MNIST

From this data generating process we obtain a rotated version of the MNIST dataset, i.e. the underlying dataset is regular MNIST and the ground truth transformations are rotation angles, $\theta_{\text{true}}(I) = r_{\text{true}}(I)$. See Fig. 4, top right panel for example data.

Our CNN obtains 99.4% test accuracy on MNIST and 41.2% on rotated MNIST (frozen weights, no re-training). The STN and P-STN ($S = 10$ training samples, $w = 3 \cdot 10^{-5}$) both learn to pose-normalise, i.e. to recover these transformations to a satisfactory degree. We compute the expected average transformation error on the $N = 10k$ rotated MNIST test images as

$$\varepsilon = \frac{1}{N} \sum_{i=1}^{N} \|\theta_{\text{true}}(I_i) - \mu(I_i)\| \mod \pi. \quad (18)$$

We get $\varepsilon = 0.76$ for the STN and $\varepsilon = 0.59$ for the P-STN.

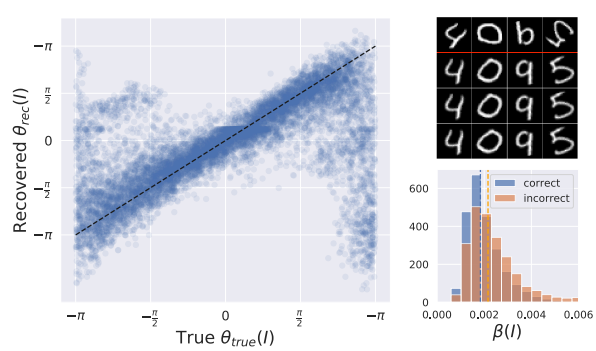

Figure 4: Rotated MNIST experiment. *Left panel:* Ground truth transformation (rotation angles in radians) against recovered transformations (mean). *Top right:* Example images from the data set and samples from the P-STN localiser. The localiser learns to pose-normalise. *Bottom right:* Outputs of the variance network. When the transformation recovery is poor (the error ε is above the median, in orange) the variances are slightly higher than when the localisation works well (blue).

Figure 6: The P-STN learns to localise traffic signs in the challenging MTSD dataset. At test time, we sample 10 transformations as shown with the various bounding boxes overlaid the images. These learned variations improves the final classification.

	Acc. ↑	NLL ↓
CNN	76.0	0.49
STN	90.6	0.31
P-STN	**92.2**	**0.29**

Table 1: Accuracy (Acc.) and negative log-likelihood (NLL) for CNN, STN and P-STN.

Uncertainty. The bottom right panel of Fig. 4 shows a histogram of $\beta(I)$, i.e. the localiser variance (or, correspondingly, the magnitude of augmentation) per image. In orange, we plot the count for images where pose-normalisation is difficult (the transformation error ε is larger than the median). In blue, we plot the variances for images that are correctly pose-normalised (transformation error ε smaller than the median). The poorly localised images are, on average, assigned 17% larger variances $\beta(I)$. The localiser uncertainty and thus the amount of data augmentation applied is somewhat meaningful, corresponding to the difficulty of the task.

5.1.2 Random placement FashionMNIST

We repeat a similar experiment on the slightly more challenging FashionMNIST dataset [Xiao et al., 2017] . The CNN baseline accuracy is 90.63%. We then randomly sample an x and y coordinate and place the FashionMNIST accordingly on a black background, after downscaling it by 50%. No rotation is applied, i.e. $\theta_{\text{true}} = [0, 0.5, t^x_{\text{true}}, t^y_{\text{true}}]$

Figure 7: Random Placement Fashion MNIST. Input images (left) and transformed samples $T_{\theta_s}(I)$ as learned by the P-STN. The P-STN learns to correctly pose-normalise and zoom in to the relevant part of the image. The samples look like plausible candidates for a data augmentation scheme, this we will explore in Sec. 5.2.

Like in previously, both localisers successfully recover θ_{true}, with the P-STN ($S = 10$ training samples, $w = 3e-05$) doing slightly better than its deterministic counterpart: test accuracies are 84.99% and 84.41%, respectively. Inspecting the transformation posterior and the resulting samples $T_{\theta_s}(I_i)$ we find that those look visually pleasing, and, as hypothesised, might be promising candidates for a data augmentation scheme. We will explore this in Sec. 5.2.

5.1.3 Mapillary street signs

Detection and classification of objects in images have many applications, e.g. for autonomous vehicles traffic signs detection is crucial. We compare a top performing classifier, a STN and our P-STN on the challenging Mapillary Traffic Sign Dataset (MTSD) [Ertler et al., 2019].

To focus this comparison, we select images that contain only one traffic sign. We obtain this subset by selecting all bounding boxes that do not intersect with other bounding boxes plus a margin of 150 px to each side. We further select the ten most common classes from this subset. This gives us a training set of 4698 images and test set of 500 images. Figure 6 shows examples images from the chosen subset.

Our classifier is a ResNet18 pre-trained on ImageNet, where we replace the last fully connected layer. We use the same resnet for the localisers in the STN and P-STN, where we similarly replace the last layer. As before, we wish to study the behavior of the localisers. Therefore, we again start by training a classifier on the ground-truth bounding boxes. We then initialise the classifier module of the STN and P-STN with this pre-trained classifier and freeze the weights of the classifier. We train the localisers of the STN and P-STN for 60 epochs with learning rate 10^{-4} and kl weight $w = 10^{-7}$. Figure 6 shows that the P-STN learns to localise the traffic signs. At test time, we sample 10 transformations illustrated by the multiple overlaying bounding boxes.

Table 1 shows that both the STN and P-STN clearly outperform the baseline classifier when trained on the full images. Even though, the STN and P-STN have exactly the same classifier, the P-STN achieves better performance because of the ensemble of classified transformations.

5.2 MARGINALISING TRANSFORMATIONS IMPROVES CLASSIFICATION ACCURACY

We have argued that marginalising transformations via samples corresponds to learned, localised data augmentations (the samples $T_{\theta_s}(I)$). We will now investigate whether these

	MNIST30	MNIST100	MNIST1000	MNIST3000	MNIST10000
CNN	70.12 ± 2.46	87.29 ± 0.58	95.80 ± 0.33	**97.48** ± 0.21	**97.82** ± 0.34 -
affine STN	69.26 ± 4.53	82.16 ± 2.30	92.05 ± 0.58	94.71 ± 0.22	96.96 ± 0.20
affine P-STN	**81.00** ± 3.92	**92.70** ± 0.74	**96.62** ± 0.58	**97.33** ± 0.17	**97.63** ± 0.23
optimal w	0.001	0.0003	0.0001	0.00003	0.00001

Table 2: The performance of a CNN, STN and P-STN on differently sized MNIST datasets.

	CNN	STN	P-STN
FaceAll	80.83 ± 0.62	82.28 ± 0.42	**84.31** ± 0.75
TwoPatterns	97.92 ± 0.53	99.79 ± 0.04	**99.96** ± 0.04
wafer	**99.63** ± 0.05	99.18 ± 0.17 -	98.86 ± 0.20
uWaveGestureLib.*	74.15 ± 1.27	79.77 ± 0.42 -	**81.13** ± 0.46
PhalangesOutlC.**	79.88 ± 1.32	**82.26** ± 0.98	81.66 ± 0.59
Mean	86.48	88.65	**89.18**

Table 3: Accuracies on a subset of the UCR timeseries dataset (full dataset names are *uWaveGestureLibrary and **PhalangesOutlinesCorrect). ± 1 STD is reported after 5 repetitions.

augmentations are indeed helpful in the downstream task, i.e. whether they improve classification performance.

5.2.1 MNIST and subsets

We compare the performance of our P-STN against a standard convolutional neural network (CNN) and a regular STN on MNIST. The standard MNIST images are centered and pose-normalised, so the localisation task is easy. Improved classifier performance can thus be viewed as an indicator for having learned an useful data augmentation scheme.

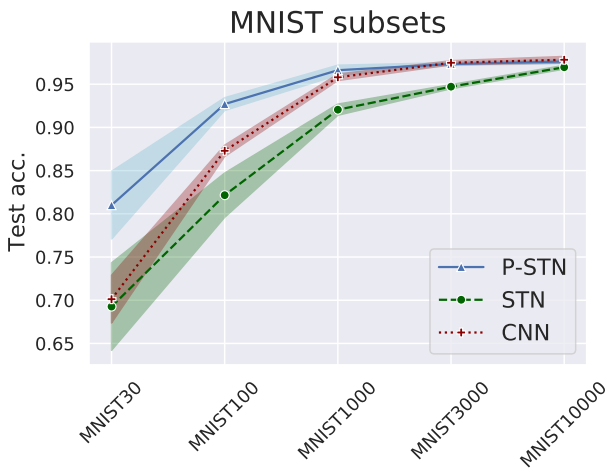

Figure 8: Performances of P-STN, STN and CNN on MNIST subsets (mean $\pm$ one STD across five folds).

Data augmentation is particularly important when training data is scarce, so we evaluate the models on small subsets of the MNIST: MNIST30 contains 30 images (i.e. 3 per class), MNIST100, MNIST1000, MNIST3000 and MNIST10000. STN and P-STN parametrise affine transformations, i.e. the learned θ is interpreted as the full affine matrix as described in Sec. 3. All models have roughly 28k parameters. We use the Adam optimiser with weight decay 0.01 and the default parameters of its PyTorch implementation. The images are color normalised. We repeat the experiment 5 times, each time with a different k-image subset of the MNIST dataset, and we report $\pm$ one standard deviation in tables and error bars. From Table 2 and Fig. 8, we see that the P-STN outperforms both the STN and CNN on the small dataset sizes. For the larger datasets the differences vanish. This supports our hypothesis: data augmentation is especially useful when data is a limited resource. This intuition is also supported by the optimal KL-weights (Table 2, bottom row) that we determine via grid search on validation data. For smaller datasets, larger w and thus more regularisation towards the variance prior (away from 0) are beneficial.

The fact that the STN performs less well than the standard CNN on this data set might be explained by the fact that the images are already nearly perfectly pose-normalised, and wrong transformations can be detrimental.

5.2.2 UCR time-series dataset

For some data modalities, such as time-series, it is not trivial to craft an useful data augmentation scheme. In this experiment, we show that the P-STN can learn an useful, non-trivial data augmentation scheme that increases performance compared to a standard on time-series data. The UCR dataset [Dau et al., 2018] is composed of 108 smaller datasets, where each dataset contains univariate time-series. The FordA dataset, for example, contains measurements of

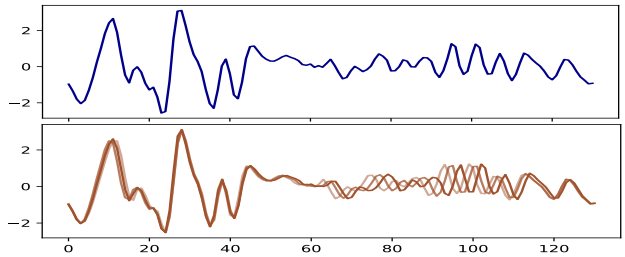

Figure 9: Examples of augmentations for a time-series from the FaceAll dataset. The top plot shows the original time-series and the bottom plot shows three augmented versions of the time-series.

engine noise over time and the goal is to classify whether or not the car is faulty. We select 5 of those subsets, each large enough to divide into training and validation set ($75/25\%$), which we use to find the optimal w via grid-search; those are $[0.0001, 1e-05, 0.001, 0.0, 0.0001]$. We draw $S = 10$ training samples. The test-set is pre-defined by the dataset curators. Learning rate and optimiser are the same as in Sec. 5.2.1, but we do not perform normalisation. All models have approximately one million parameters. Table 3 shows that the P-STN achieves higher mean accuracy than both the STN and the CNN, indicating that we can automatically learn an useful data augmentation scheme for time-series.

We verify this qualitatively in Fig. 9 which shows an example of the learned data augmentation. We see that the model does not simply apply a global transformation, but learns to augment the time-series more in some intervals, such as in $[60; 110]$, and augment the time-series less in other intervals, such as in $[0; 50]$.

5.3 MARGINALISING TRANSFORMATIONS IMPROVES CALIBRATION

In Sec. 5.1 we have seen that harder images on average have larger transformation uncertainties. We now investigate if those meaningful localisation uncertainties translate into meaningful uncertainties in the downstream task, i.e. the calibration of our classifier. At test-time, we evaluate

$$p(y|I) = \int p(y|I, \theta) q(\theta) \mathrm{d}\theta \approx \frac{1}{S} \sum_{s=1}^{S} p(y|T_{\theta_s}(I)), \quad (19)$$

i.e. we will investigate how well the uncertainty in this distribution matches the quality of predictions. Fig. 10 shows a calibration plot for the MNIST100 subset classification task from Sec. 5.2.1 for the CNN, STN and P-STN for two different w-parameters; $w = 0.0003$ yields the best performance (reported in Table 2) and $w = 0.0001$ yields the best calibration. The expected calibration errors [Guo et al., 2017, Küppers et al., 2020, 2021] are CNN: 0.0743 ± 0.0094, STN: 0.1160 ± 0.0205, P-STN, $w = 0.0003$ (optimal performance model): 0.0567 ± 0.0065, P-STN, $w = 0.0001$ (optimal calibration model): $\mathbf{0.0271} \pm 0.0088$. We report the mean over 5 folds, $\pm$ one STD. The P-STN significantly improves calibration in the downstream classification task.

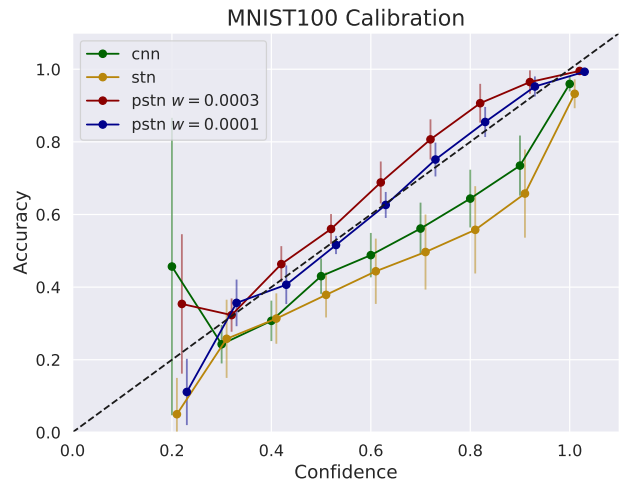

Figure 10: Calibration plots for CNN, STN and two P-STN models. One with KL-weight yielding optimal performance ($w = 0.0003$) and one with KL-weight yielding optimal calibration ($w = 0.0001$). Both P-STN models are better calibrated than CNN and STN.

6 CONCLUSION

We have introduced a probabilistic extension to the spatial transformer network (STN) [Jaderberg et al., 2015]. Our work took motivation from the empirical observation that the STN is often brittle to train as a poorly predicted transformation may prevent the model from getting any gradient signal, resulting in divergent optimization. Our probabilistic STN (P-STN) instead approximates the posterior distribution of transformations using amortised variational inference, and marginalises accordingly. As is common, marginalisation improves the robustness of the model.

Empirically, we, in particular, note two advantages of the probabilistic formulation over the deterministic. First, the performance of the localisation network is improved, since the Monte Carlo marginalisation effectively amounts to trying many different transformations. Secondly, the probabilistic formulation improves the overall model performance, since the sampled transformations act as data augmentation both during training and during testing. The resulting ensemble of predictions is more accurate and better calibrated than common classifiers as well as the original spatial transformer.

References

Henry S Baird. Document image defect models. In *SDIA*, pages 546–556. Springer, 1992.

Gregory Benton, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. Learning invariances in neural networks. *arXiv preprint arXiv:2010.11882*, 2020.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *ArXiv*, abs/1601.00670, 2017.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. The ucr time series classification archive, October 2018. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Nicki Skafte Detlefsen. libcpab. `https://github.com/SkafteNicki/libcpab`, 2018.

Nicki Skafte Detlefsen and Søren Hauberg. Explicit disentanglement of appearance and perspective in generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Nicki Skafte Detlefsen, Oren Freifeld, and Søren Hauberg. Deep diffeomorphic transformer networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4403–4412, June 2018.

Nicki Skafte Detlefsen, Martin Jørgensen, and Søren Hauberg. Reliable training and estimation of variance networks. In *33rd Conference on Neural Information Processing Systems*, 2019.

Christian Ertler, Jerneja Mislej, Tobias Ollmann, Lorenzo Porzi, and Yubin Kuang. Traffic sign detection and classification around the world. *CoRR*, abs/1909.04422, 2019. URL `http://arxiv.org/abs/1909.04422`.

Oren Freifeld, Søren Hauberg, Kayhan Batmanghelich, and John W. Fisher. Transformations based on continuous piecewise-affine velocity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Ian Goodfellow, Honglak Lee, Quoc V Le, Andrew Saxe, and Andrew Y Ng. Measuring invariances in deep networks. In *NIPS*, pages 646–654, 2009.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schon. Evaluating scalable bayesian deep learning methods for robust computer vision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 318–319, 2020.

Søren Hauberg, Oren Freifeld, Anders Boesen Lindbo Larsen, John W. Fisher, and Lars Kai Hansen. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. In *Artificial Intelligence and Statistics*, pages 342–350, 2016.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

Angjoo Kanazawa, David W Jacobs, and Manmohan Chandraker. Warpnet: Weakly supervised matching for single-view reconstruction. In *CVPR*, 2016.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL `http://arxiv.org/abs/1312.6114`.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural

networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

Fabian Küppers, Jan Kronenberger, Amirhossein Shantia, and Anselm Haselhoff. Multivariate confidence calibration for object detection. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

Fabian Küppers, Jan Kronenberger, Jonas Schneider, and Anselm Haselhoff. Bayesian confidence calibration for epistemic uncertainty modelling. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, July 2021.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

Yann LeCun, LD Jackel, Leon Bottou, A Brunot, Corinna Cortes, JS Denker, Harris Drucker, I Guyon, UA Muller, Eduard Sackinger, et al. Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks*, volume 60, pages 53–60. Perth, Australia, 1995.

Chen-Hsuan Lin and Simon Lucey. Inverse compositional spatial transformer networks. *CoRR*, abs/1612.03897, 2016. URL `http://arxiv.org/abs/1612.03897`.

Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Training invariant support vector machines using selective sampling. *Large scale kernel machines*, pages 301–320, 2007.

David J C MacKay. Model comparison and Occam's razor. *Information Theory, Inference and Learning Algorithms*, pages 343–355, 2003.

David JC MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992.

Pola Schwöbel, Martin Jørgensen, Sebastian W. Ober, and Mark van der Wilk. Last layer marginal likelihood for invariance learning, 2021. URL `https://arxiv.org/abs/2106.07512`.

Ron A Shapira Weber, Matan Eyal, Nicki Skafte, Oren Shriki, and Oren Freifeld. Diffeomorphic temporal alignment nets. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 6570–6581. 2019.

Patrice Y Simard, Dave Steinkraus, and John C Platt. Best practices for convolutional neural networks applied to visual document analysis. In *2013 12th International Conference on Document Analysis and Recognition*, volume 2, pages 958–958. IEEE Computer Society, 2003.

Søren Kaae Sønderby, Casper Kaae Sønderby, Lars Maaløe, and Ole Winther. Recurrent spatial transformer networks. *arXiv preprint arXiv:1509.05329*, 2015.

Andrew Stirn and David A Knowles. Variational variance: Simple and reliable predictive variance parameterization. *arXiv e-prints*, pages arXiv–2006, 2020.

Hiroshi Takahashi, Tomoharu Iwata, Yuki Yamanaka, Masanori Yamada, and Satoshi Yagi. Student-t variational autoencoder for robust density estimation. In *IJCAI*, pages 2696–2702, 2018.

Mark Van der Wilk, Matthias Bauer, ST John, and James Hensman. Learning invariances using the marginal likelihood. In *Advances in Neural Information Processing Systems*, pages 9938–9948, 2018.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.