CS236 Project Progress Report
Neural Processes for Image Completion

Amaury Sabran asabran@stanford.edu
Arnaud Autef arnaud15@stanford.edu
Benjamin Petit benpetit@stanford.edu

Stanford University

Abstract

In this project, we develop image completion techniques based on Neural Processes (1; 2), a recently proposed class of models that uses neural networks to describe distributions over functions. We show that the Neural Process model seamlessly applies to the problem of image completion, explore different approaches to this task and discuss their performance on two well-known datasets: MNIST (5) and CIFAR10 (8).

1 Introduction

Image completion is a common problem in image processing, which can be stated as follows: given a subset of an image's pixels, how can we infer the missing pixels to reconstruct the entire image? Image completion techniques can be used in many situations, such as *image cleaning*, *object removal* and *image super-resolution*.

However, the problem is ill-defined: given a number of known pixels, several completions of the image are possible. To tackle it, two broad types of approach have proved successful:

- *Use a penalization* (such as the total variation distance) to find the most "simple" image which satisfies the set of constraints represented by the known pixels.
- *Learn a probability distribution* over "natural" images from a training set of images. Then partial images are completed with pixels that make it very likely under the distribution learned.

In this report, we present an approach that falls in the second category. More specifically, we build up on a new framework called **Neural Processes - NPs**. NPs can efficiently infer a latent representation of an image based on a subset of its pixels. This latent representation can then be used to sample completed images.

2 Related work

Previous approaches to **image completion** problems include PixelCNN (4) and PixelRNN (3). PixelRNN is an autoregressive model which uses a recurrent neural network - RNN - to model the distribution of a pixel's color given an encoding of previous pixel values (in raster scan order). PixelCNN uses a similar autoregressive approach, but uses masked convolutions instead of a RNN.

Because of their autoregressive nature, PixelRNN and PixelCNN are constrained to specific configurations of the missing pixels. For example, most variants of PixelRNN and PixelCNN assume that the first k rows of pixels are known. NPs are appropriate for any configuration of observed

Preprint. Work in progress.

pixels.

Neural Processes (NPs) were introduced (1; 2) as a bridge between deep Neural Networks - NNs - and Gaussian Processes - GPs. NNs are powerful and scalable function approximators, but lack the ability to represent a class of functions. NPs extend the idea of distributions over classes of functions to deep NNs.

Gaussian Processes (GPs) encode distributions over classes of functions. Given a number of observations, one can update a posterior distribution over a class of possible functions, making it possible to give estimates of the value of unseen points, as well as confidence intervals. GPs are data-efficient and robust. However, they:

- Lack the expressiveness of deep neural networks.
- Do not scale well to high dimensional problems.
- Are computationally expensive.

Current attempts to bridge this gap include *Deep GPs*, that stack GPs to obtain deep models (12); *Variational Implicit Processes* (11), conceptually similar to NPs but using GPs instead of NNs to learn an implicit stochastic process; *Matching Networks* (10) and *Deep Kernel Learning* (9), which use NNs to extract representations from the data but pass this representation to an explicit distance kernel.

NPs are an immediate extension to a previous paper by Garnelo and al. (1), where they introduces *Conditional Neural Processes* - CNPs - a slightly less general model. NPs are to CNPs what *Variational Autoencoders* - VAEs - are to *Autoencoders*: CNPs do not introduce randomness in the latent space while NPs parameterize a distribution over latent variables, making them untractable to optimize via maximum likelihood. Therefore, NPs training is similar to VAEs and involves maximizing a variational lower bound on the log likelihood.

3 Problem statement:

In **image completion** tasks, images are commonly represented by $3D$ tensors:

$$(A_{i,j,k})_{1\leq i\leq H, 1\leq j\leq W, 1\leq k\leq C}$$

Where the first two dimensions represent the x and y coordinates of the pixels on a width W and height H image, and the third dimension corresponds to color channels (3 channels are used for RGB images, a single one for grayscale images). However, as mentioned above, NPs allow us to learn a probability distribution over a class of *functions*. Therefore, we view images as a function $f : [0, 1]^2 \to \mathbb{R}^C$, such that:

$$\forall 1 \leq i \leq H, 1 \leq j \leq W, 1 \leq k \leq C,\ f(\frac{i}{H}, \frac{j}{W})_k = A_{i,j,k}$$

This framework allows us to define the image completion problem as follows: given an unknown image (function) f, a number of pixel coordinates (context points) $x_1, ..., x_n \in [0, 1]^2$ and pixels values $y_1 = f(x_1), ..., y_n = f(x_n)$, infer a representation $\hat{f}$ of f which can be used to compute the color of all other pixels. More specifically, we wish to be able to recover missing pixels $\tilde{y}$ of coordinates $\tilde{x}$, by sampling them from a posterior distribution over $\mathbb{R}^C$ given the observations $(x_i, y_i)_{1\leq i\leq n}$:

$$\forall \tilde{x} \in [0, 1]^2,\ \tilde{y} \sim p_\theta(\tilde{x}|(x_i, y_i)_{1\leq i\leq n})$$

In our approach, the **generative model** used to learn the posterior probability distribution $p_\theta(.|(x_i, y_i)_{1\leq i\leq n})$ is a **NP**.

4 Approach

4.1 The Neural Process Model

A NP is a generative model that learns a probability distribution $\mathbb{P}(f)$ over a set of functions $F = \{f : \mathcal{X} \to \mathcal{Y}\}$, using datasets $D_f = \{(x_i, y_i)_{1\leq i\leq n},\ x_i \in \mathcal{X}, y_i \in \mathcal{Y}\}$ with $\forall i, y_i = f(x_i)$ and

$f \in F$. The different datasets should illustrate the variability of functions f in F. The probability of observations $(y_i)_{1 \leq i \leq n}$, conditionally on context points $(x_i)_{1 \leq i \leq n}$ is then:

$$\rho(y_{1:n}|x_{1:n}) = \mathbb{E}_{f \sim \mathcal{P}(f)} \left[p(y_{1:n}|f, x_{1:n}) \right]$$

Where $p(y_{1:n}|f, x_{1:n})$ is modeled with *two* assumptions. First, we assume the *independence* of observations $(y_i)_{1 \leq i \leq n}$, conditionally on context points $(x_i)_{1 \leq i \leq n}$:

$$p(y_{1:n}|f, x_{1:n}) := \prod_{i=1}^{n} p_{\theta(x_i, f)}(y_i)$$

Then, we introduce *observation noise*, meaning that: $p_{\theta(x_i,f)}(y_i) \neq 1_{y_i = f(x_i)}$. In practice, $p_{\theta(x_i,f)}(y_i)$ is a normal distribution when the space $\mathcal{Y}$ is continuous or a Bernoulli distribution for binary outputs $\mathcal{Y} = \{0, 1\}$.

With NPs, we then make the assumption that the associated stochastic process can be parameterized by a high dimensional latent vector $z \in \mathbb{R}^p$ and a **decoder** neural network g:

$$p_{\theta(x_i,f)}(y_i) \approx p_{g(x_i,z)}(y_i)$$

Giving:

$$p(y_{1:n}|x_{1:n}, z) \approx \prod_{i=1}^{n} p_{g(x_i,z)}(y_i)$$

Where we use a variational posterior of the latent variables $q(z|x_{1:n}, y_{1:n})$ to learn g. This variational posterior is parameterised by another **encoder** neural network h that is invariant to permutations.

4.2 Evidence Lower Bound

We can use the expression of the joint distribution of samples given a latent context z to derive an Evidence Lower Bound - ELBO - for their likelihood. n points $(x_i)_{1 \leq i \leq n}$ are provided, the first m, $(x_i)_{1 \leq i \leq m}$, $m < n$, are context points and are associated with observed values $(y_i)_{1 \leq i \leq m}$. Thus, the ELBO for the generated, unobserved, values $(y_i)_{m+1 \leq i \leq n}$ is (full derivation of the ELBO is exposed in appendix):

$$\log p(y_{m+1:n}|x_{1:n}, y_{1:m}) \geq \mathbb{E}_{z \sim q(z|x_{1:n}, y_{1:n})} \left[\log \frac{p(z|x_{1:m}, y_{1:m})}{q(z|x_{1:n}, y_{1:n}} + \sum_{i=m+1}^{n} \log p(y_i|z, x_i) \right]$$

However, as the conditional prior $p(z|x_{1:m}, y_{1:m})$ is untractable, it is *approximated* by $q(z|x_{1:m}, y_{1:m})$ and the training objective becomes:

$$\boxed{\mathbb{E}_{z \sim q(z|x_{1:n}, y_{1:n})} \left[\log \frac{q(z|x_{1:m}, y_{1:m})}{q(z|x_{1:n}, y_{1:n})} + \sum_{i=m+1}^{n} \log p(y_i|z, x_i) \right]}$$

4.3 Architecture

The model and the loss function have been defined above, and lead to the following architecture:

1. An **encoder network** h takes context points from one of our datasets $D_f = (x_i, y_i)_{1 \leq i \leq n}$ and embeds them to *context* vectors of fixed dimension $r_i = h(x_i, y_i) \in \mathbb{R}^p$.
2. An **aggregator** a aggregates context points embeddings r_i into a single embedding of the full context r. For our architecture to model a stochastic process, the aggregator a must be *order invariant*: $\forall n \in \mathbb{N},\ \forall \sigma \in \mathfrak{S}_\mathfrak{n},\ a\left(r_1, ..., r_n\right) = a\left(r_{\sigma(1)}, ..., r_{\sigma(n)}\right)$. In (2) and our first experiments, a simple mean aggregator is taken:

$$a\left(r_1, ..., r_n\right) = \frac{1}{n} \sum_{i=1}^{n} r_i$$

3. A **context to distribution** network links aggregated contexts r to the mean and variance parameters of a normal distribution over the latent space $z \sim \mathcal{N}(\mu(r), I\sigma^2(r))$.

4. A **conditional decoder** network $g(z, x)$ maps any point $x \in \mathcal{X}$ paired with a latent context $z \in \mathbb{R}^p$ to a distribution over the output space $\mathcal{Y}$.

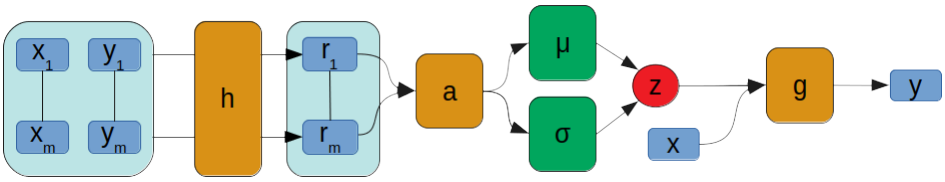

Figure 1: **Overview of the NP architecture**: h is the encoder, a is the aggregator, μ and σ are the parameters of the normal distribution from which the latent context z is sampled, and g is the decoder

4.4 Architecture improvement: attention model for aggregation

In our latest experiments, we experimented the use of an **attention model** (7) instead of a simple mean aggregation, which we thought was introducing to much *regularization* in the architecture. We replace the mean aggregation:

$$a(r_1, ..., r_n) = \frac{1}{n} \sum_{i=1}^{n} r_i$$

By the following function:

$$a(r_1, ..., r_n) = \sum_{i=1}^{n} \alpha_i r_i$$

Where

$$\alpha_i = \frac{\exp w^T r_i}{\sum_{k=1}^{n} \exp w^T r_k}$$

With trainable attention weight w. Note that this aggregation method is still permutation invariant, which is essential for the NP model to represent stochastic processes as described in section **4.1**. Intuitively, the ordering of observations must not have an influence on our prior, our representation of the function space F comes from observations observed "simultaneously".

4.5 Application to the image completion problem

Images are described by a function $f : [0, 1]^2 \to \mathbb{R}^C$. We can encode any (pixel, color) pair as a variable x and the corresponding value (color) $y = f(x)$, creating an observation (x, y) that we can feed to the NP. Actual training steps are then the following:

1. Sample a batch of B images from the dataset.
2. For each image in the batch, sample m context points $(x_i, y_i)_{1 \leq i \leq m}$. x_i are the pixel coordinates, y_i the pixel values, m is sampled from an uniform distribution between 1 and the total number of pixels of an image n.
3. The training objective is an approximation of the NELBO:

$$-\mathbb{E}_{z \sim q(z|x_{1:n}, y_{1:n})} \left[\sum_{i=m+1}^{n} \log p(y_i|z, x_i) + \log \frac{q(z|x_{1:m}, y_{1:m})}{q(z|x_{1:n}, y_{1:n})} \right]$$

4. Sample the latent variable $z \sim q(z|x_{1:n}, y_{1:n})$ and compute the two terms using the sampled context $(x_i, y_i)_{1 \leq i \leq m}$ to get a Monte-Carlo estimate of the training objective
5. Update the parameters of our networks via gradient descent

5 Experiments

5.1 Experimental setting

We will evaluate our approach on two well-known **datasets**:

- **Binary MNIST** (5), which consists in roughly $70,000$ 28×28 black-and-white images of handwritten digits.
- **CIFAR-10** (8), which consists in roughly $60,000$ 32×32 (tri-channel) tiny images of fairly complex objects belonging to 10 non overlapping classes. Each image belongs to one, and only one, of the following categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

We expect image completion to be much more challenging on CIFAR-10, as it contains tri-channel images with much more complex structure. We first planned to carry out our experiments on **CelebA**, a famous dataset of celebrity faces. We decided to use CIFAR instead, it is a more reasonable "next step" after experiments on Binary MNIST: images structure is more complex but still limited to 10 non overlapping classes.

To analyze our results, we use two types of **metrics**: when comparing variants of our NP model on a dataset, we compare the **losses** achieved by each variant **on the test set**; when comparing context point selection strategies for image completion, we compare the **average likelihood of pixels** of the true image according to the probability distribution obtained by conditioning on selected context points.

We implemented our NP models in Python, using the *Pytorch* (13) deep learning framework. Our implementation is available on the following GitHub repository: `https://github.com/Arnaud15/CS236_Neural_Processes_For_Image_Completion`. Computations were carried out on a Tesla V100 GPU and took times ranging from 1 to ~ 12 hours.

5.2 Binary MNIST

5.2.1 Architecture details

We chose the following architectures for the encoder and decoder:

- Encoder h : MLP with 2 hidden layers, 200 hidden units per layer and 128 output units
- Context to distribution (μ, σ): we used linear layers for μ and σ and a latent space with 128 dimensions for $z \sim \mathcal{N}(\mu, \sigma)$.
- Decoder g : MLP with 4 hidden layers and 200 hidden units per layer

We trained our network using the Adam (6) optimizer with a learning rate of 10^{-3}, for 350 epochs.

5.2.2 Results

We performed **image-completion experiments** with images from the test set of MNIST. For each masked input image we sample a context representation z and use the decoder to reconstruct the whole image $g(z)$. The results for **100 context pixels** are presented in Figure 2. The first row displays the original images, the second row indicates the context pixels (missing pixels are in blue) and the following rows are sampled reconstructed images.

We also performed reconstruction after **hiding half of the image**. We notice that different sampled latent variables z can yield very dissimilar reconstructed digits (seventh column of the right image).

Model	NP w/ mean agg.	NP w/ attention agg.
Loss (train)	62.88	58.58
Loss (test)	62.50	57.95

Table 1: Train/test loss, MNIST

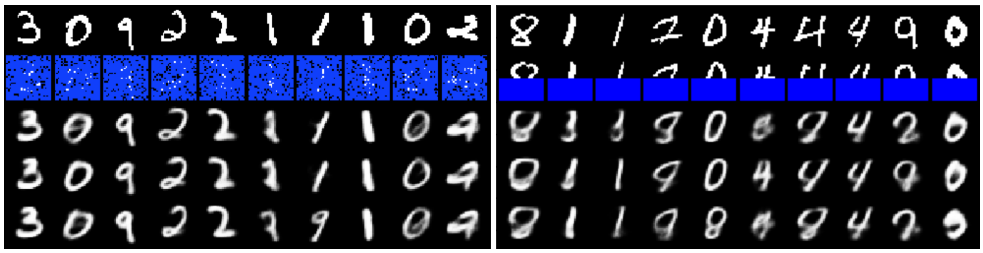

Figure 2: Sampled reconstructed images with masked inputs

We compared the **performance of the NP model with mean and attention-based aggregations**. We report the train and test loss of boths model after convergence in Table 1. The model with attention-based aggregation gives much better results in terms of test loss. This visually translates into sharper images and less blurry contours, as can be seen in Figure 3.

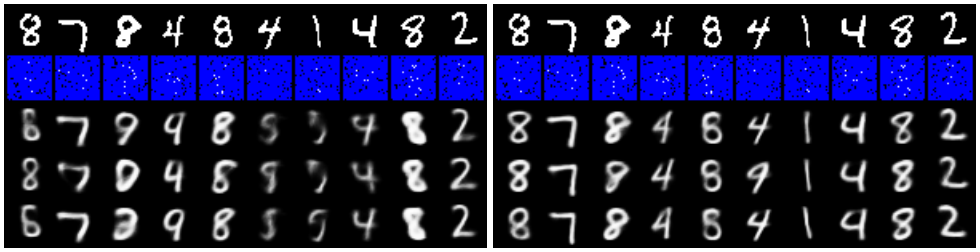

Figure 3: Sampled reconstructed images, 50 context pixels, with and without attention (left/right)

Finally, we explored several **context selection heuristics** for image completion. For a given budget of N context points with paired with observed pixel values, several strategies can be thought of to select the best N points, in an active learning fashion. Note that we expect our heuristics to provide a mere approximation of the best solution, as the *best subset selection problem* is intrinsically intractable. Below are the strategies we investigated:

- **Random** context: observe n pixels at random.
- **Sequential** context: observe the first n pixels in raster ordering.
- **Greedy highest variance** context: sequentially request, as the next observed pixel, the point where the predicted distribution over pixel values has the highest variance.

Evaluation of context selection heuristics: We evaluated these strategies on Binary MNIST. For each strategy, we sampled images from the test set, and evaluated the likelihood of the true image (in terms of average pixel likelihood) given contexts of increasing sizes. Results are presented in Figure 4. The highest variance heuristic clearly outperforms the two other techniques. This can be explained as follows: the highest variance heuristic requests pixels for which the current context has the most uncertainty, which allows it to break "ties" between equally likely images very efficiently after very few pixels.

A randomly sampled context also gives a quite good performance (even though it remains much worse than the highest variance heuristic), especially compared to a sequential context, which reads the pixels in raster ordering. This can be explained by the fact that the first rows of MNIST images

do not contain much useful information (they are only composed of black pixels), whereas sampling a sufficient number of random pixels in an image yields information from various parts of the image. This can be seen graphically: the average pixel likelihood for the sequential context tends to plateau at the beginning (first rows contain very little information), whereas that of the random context smoothly increases from the first pixel.

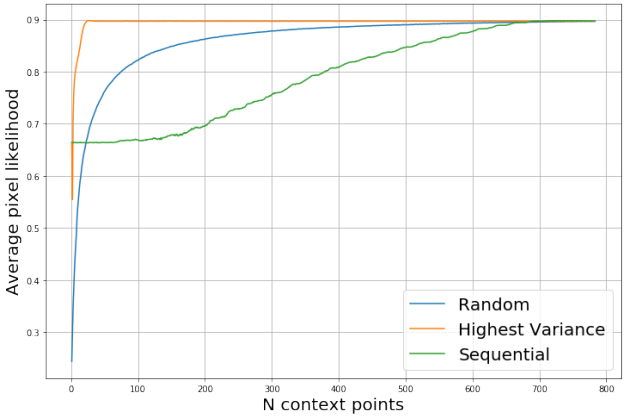

Figure 4: Comparison of image completion strategies on MNIST

5.3 CIFAR-10

5.3.1 Architecture details

The architecture that we used is given below. Notice that we used deeper and wider neural networks than on Binary MNIST. The outputs are also slightly different, as the image's pixels are not binary variables anymore. Instead, we parameterize them as multivariate Gaussian distributions over the three color channels (with a diagonal covariance matrix).

- Encoder h : MLP with 3 hidden layers, 400 hidden units per layer and 400 output units
- Context to distribution (μ, σ): we used linear layers for μ and σ and a latent space with 400 dimensions for $z \sim \mathcal{N}(\mu, \sigma)$.
- Decoder g : MLP with 4 hidden layers and 400 hidden units per layer.

We trained our network using the Adam (6) optimizer with a learning rate of 10^{-3}, for 500 epochs.

5.3.2 Results

Figure 5: Sampled reconstructed images, 50 context pixels, left : with attention, right : without attention. Top row is the ground truth, second row is the context (blue pixels are hidden), subsequent rows alternate between mean and variance estimates of the color of each pixel for different sampled z.

Model	NP w/ mean agg.	NP w/ attention agg.
Loss (train)	1,010.0	976.9
Loss (test)	1,017.0	991.5

Table 2: Train/test loss, CIFAR-10

Figure 5 presents a few examples of **image completion on CIFAR-10**. The context is given by 50 randomly sampled pixels. The images are blurry, however they always represent natural objects and contours, even given very few context pixels (it would be hard for a human to even guess what the hidden image represents).

Figure 6: Sampled reconstructed images, 450 context pixels, with and without attention (left/right)

Figure 6 depicts the same situation, with a lot more context pixels (450 random pixels for each image). The images remain blurry even with the attention-based aggregation. However, one can notice that there is much less variance in the estimates: different sampled latent variables yield very similar images, which is in line with our intuition. Indeed, there is much more uncertainty on what the image actually represents with a context of size 50 than with 450 context pixels.

Discussion: On this more complex dataset, our NP model is not able to sample images with well-defined contours. However, out of a few context points, our NP model is able to sample images for which the background / foreground limit is well identified. With more context points, uncertainty on on the foreground / background boundary goes down quickly. Finally, the benefits of our simple attention model for aggregation are much less striking on CIFAR 10, but this can be explained by our architecture choices. The encoder has much more parameters in our CIFAR10 experiment and can learn directly to set the uninformative context points (x_i, y_i) to a low-norm vector r_i: the attention mechanism doesn't help much. With a smaller encoder, the attention mechanism helps to distinguish informative and uninformative context points. With a richer attention model on CIFAR10 (a full NN instead of a single trainable vector), we expect that the benefits of attention would have been more noticeable.

6 Conclusion

Throughout this project, our team has been able to:

- Implement Neural Processes, and apply them to image completion on two popular image processing datasets: **Binary MNIST and CIFAR10**.
- Show the benefits of using an **attention model aggregator** in a NP, when looking at the test loss of the model. We also noticed qualitative improvements in terms of samples quality (sharper details), and the diversity of sampled images from a given set of context points.
- Evaluate several **heuristics for the selection of context points**, with a fixed budget of N observations. Such heuristics could be useful when applying NPs to **black-box optimization**, where we want to have a good guess on the maximum of a function f out of as few queries of f as possible.

Now, given our results, the following next steps are worth considering:

- To try out more **complex encoder and attention models on CIFAR10** and quantify the benefits they bring in terms of test loss.
- On MNIST, from our NP model with attention we were able to sample more diverse digits from a fixed context of observations: attention may help avoiding **mode collapse** issues with NPs, and this intuition should be assessed by *ad hoc* experiments. On CIFAR, we observe mode collapse and different sampled z always give the same reconstructed image.
- The images obtained on CIFAR10 are very blurry. We suspect the **reconstruction term** of our loss (which corresponds to a mean squared error with our gaussian model on pixel values) to be responsible for this issue and there may be room for improvement here.

Overall, NPs are a very flexible class of models, we enjoyed analyzing their performance in the context of image completion, but many more tasks can be tought of. In a reinforcement learning environment where the agent can be in distinct game "stages", a NP could learn the environment dynamics of the different stages. Then, conditionally on past observations, an agent using this NP could figure out in which game stage it is playing and what dynamics to expect to select the best action.

Finally, our team would like to thank all the teaching staff of this Deep Generative Models class: we learned a lot on this broad topic, and were provided with a good understanding of the main families of generative models, with their advantages as well as their limitations.

References

[1] Garnelo, Marta, et al. "Conditional Neural Processes." International Conference on Machine Learning (2018).

[2] Garnelo, Marta, et al. "Neural processes." arXiv preprint arXiv:1807.01622 (2018).

[3] Van Oord, Aaron, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel Recurrent Neural Networks." International Conference on Machine Learning (2016).

[4] Van Oord, Aaron, et al. "Conditional Image Generation with PixelCNN Decoders." Advances in Neural Information Processing Systems. 2016.

[5] LeCun, Yann. "The MNIST database of handwritten digits." http://yann. lecun. com/exdb/mnist/ (1998).

[6] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[7] Ilse, Maximilian, Jakub M. Tomczak, and Max Welling. "Attention-based deep multiple instance learning." arXiv preprint arXiv:1802.04712 (2018).

[8] Alex Krizhevsky, "Learning multiple layers of features from tiny images" Technical Report, 2009.

[9] Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. "Deep kernel learning." In Artificial Intelligence and Statistics, 2016.

[10] Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. "Matching networks for one shot learning." In Advances in Neural Information Processing Systems, 2016.

[11] Ma, C., Li, Y., and Hernández-Lobato, J. M. "Variational implicit processes." arXiv preprint arXiv:1806.02390, 2018.

[12] Damianou, A. and Lawrence, N. "Deep gaussian processes." In Artificial Intelligence and Statistics, 2013.

[13] Paszke, Adam and Gross, Sam and Chintala, Soumith and Chanan, Gregory and Yang, Edward and DeVito, Zachary and Lin, Zeming and Desmaison, Alban and Antiga, Luca and Lerer, Adam. "Automatic differentiation in PyTorch." 31st Conference on Neural Information Processing Systems (NIPS), 2017.

Appendix A: derivation of the variational lower bound

The variational lower bound can be computed as follows:

$$\begin{aligned}\log p(y_{m+1:n}|x_{1:n}, y_{1:m}) &= \log p(y_{m+1:n}|x_{1:n}, y_{1:m}) \\ &= \sum_z q(z|x_{1:n}, y_{1:n}) \log \frac{p(y_{m+1:n}|x_{1:n}, y_{1:m}, z)p(z|y_{1:m}, x_{1:m})}{p(z|y_{1:n}, x_{1:n})} \\ &= \sum_z q(z|x_{1:n}, y_{1:n}) \left(\log p(y_{m+1:n}|x_{1:n}, y_{1:m}, z) + \log \frac{p(z|y_{1:m}, x_{1:m})q(z|x_{1:n}, y_{1:n})}{p(z|y_{1:n}, x_{1:n})q(z|x_{1:n}, y_{1:n})} \right)\end{aligned}$$

The first term gives:

$$\begin{aligned}\sum_z q(z|x_{1:n}, y_{1:n}) \log p(y_{m+1:n}|x_{1:n}, y_{1:m}, z) &= \sum_z q(z|x_{1:n}, y_{1:n}) \log \prod_{i=m+1}^{n} p(y_i|x_i, z) \\ &= \mathbb{E}_{z\sim q(z|x_{1:n}, y_{1:n})} \left(\sum_{i=m+1}^{n} \log p(y_i|x_i, z) \right)\end{aligned}$$

The second term gives:

$$\sum_z q(z|x_{1:n}, y_{1:n}) \log \frac{p(z|y_{1:m}, x_{1:m})q(z|x_{1:n}, y_{1:n})}{p(z|y_{1:n}, x_{1:n})q(z|x_{1:n}, y_{1:n})} =$$

$$D_{KL}(q(z|x_{1:n}, y_{1:n})||p(z|y_{1:n}, x_{1:n}) + \sum_z q(z|x_{1:n}, y_{1:n}) \log \frac{p(z|y_{1:m}, x_{1:m})}{q(z|x_{1:n}, y_{1:n})}$$

So that finally, as $D_{KL}(q(z|x_{1:n}, y_{1:n})||p(z|y_{1:n}, x_{1:n}) \geq 0$:

$$\log p(y_{m+1:n}|x_{1:n}, y_{1:m}) \geq \mathbb{E}_{z\sim q(z|x_{1:n}, y_{1:n})} \left(\sum_{i=m+1}^{n} \log p(y_i|x_i, z) + \log \frac{p(z|y_{1:m}, x_{1:m})}{q(z|x_{1:n}, y_{1:n})} \right)$$