A Gaussian Process perspective on Convolutional Neural Networks

Anastasia Borovykh[*]

Abstract

In this paper we cast the well-known convolutional neural network in a Gaussian process perspective. In this way we hope to gain additional insights into the performance of convolutional networks, in particular understand under what circumstances they tend to perform well and what assumptions are implicitly made in the network. While for feedforward networks the properties of convergence to Gaussian processes have been studied extensively, little is known about situations in which the output from a convolutional network approaches a multivariate normal distribution. In the convolutional net the sum is computed over variables which are not necessarily identically distributed, rendering the general central limit theorem useless. Nevertheless we can apply a Lyapunov-type bound on the distance between the Gaussian process and convolutional network output, and use this bound to study the properties under which the convolutional network behaves approximately like a Gaussian process, so that this behavior –depending on the application– can be either obtained or avoided.

1 Introduction

A convolutional neural network (CNNs) is a biologically-inspired type of deep neural network (DNN) that has recently gained popularity due to its success in classification problems in particular computer vision and speech recognition, see e.g. [Krizhevsky et al., 2012], [Kim, 2014], [Karpathy et al., 2014], but also in forecasting problems [van den Oord et al., 2016]. The CNN consists of a sequence of convolutional layers, the output of which is connected only to local regions in the input. This is achieved by a convolution: sliding a filter over the input and at each point computing the dot product between the two. This structure allows the model to learn filters that are able to recognize specific patterns in the input data and moreover combine the learned patterns in a hierarchical way.

While convolutional networks have been shown to possess the ability of learning invariant features, many of the inner workings of a CNN still remain unknown. Approaching the convolutional network from a Bayesian perspective by assuming a probability distribution on the network parameters one can induce a distribution on the output function. While computing the distribution of the output of the network exactly can be challenging, one can study the network by means of its limiting behavior.

It is well known that a neural network with an infinite number of hidden nodes and weights initialized from a Gaussian distribution, by an application of the central limit theorem (CLT), outputs a function with a Gaussian process prior [Neal, 2012]. This work motivated a theoretical understanding of neural networks and deep learning through the kernel methods and in particular Gaussian processes (GPs).

1.1 Our contributions

While convolutional neural networks were developed from a practice-based approach, and have empirically been shown to be very powerful, we argue that also theoretically these networks should be rationalized. We extend the theoretical understanding of CNNs by connecting convolutional neural networks to Gaussian processes. Unlike feedforward neural networks in which the output layer computes a sum of independent and identically distributed random variables, in the convolutional network the sum in each layer is computed over independent but not necessarily identically distributed weighted variables from the previous layer, thus possessing a compositional structure which the feedforward net lacks. While the general CLT is unapplicable in this non-identically distributed case, we can employ a generalized central limit

[*]Dipartimento di Matematica, Università di Bologna, Bologna, Italy.**e-mail**: anastasia.borovykh2@unibo.it

theorem [Bentkus, 2005] in the form of a Lyapunov-type bound to study the convolutional network from a Gaussian process perspective. Our work therefore extends the well-known concept of convergence to GP behavior in wide feedforward neural networks to convolutional networks. Besides the theoretical derivations we empirically study the discrepancy between convolutional networks with finite filters and the Gaussian processes in which the kernel is defined recursively. In this way we gain insight into the behaviour of the CNNs, and in particular how and when a GP behavior can be obtained or avoided. As the receptive field of a CNN grows with both filter size and number of layers, we show that using a linear activation allows for a convergence dependent on the total receptive field. Using non-linear activations however allows to deviate from the GP behavior, since it seems to no longer be possible to propagate the dependence on the input throughout the network. In this way only a growing filter width results in convergence, while a larger number of layers results in a divergence between the two.

1.2 Related work

Gaussian processes have been a well-known instrument used in forecasting time series by defining through them a flexible prior over functions in regression models [Rasmussen and Williams, 2006]. The ability of the GP to learn meaningful dependencies is fully encoded by the covariance function, a function that specifies the dependence between the inputs and learning a flexible and expressive covariance kernel is therefore an active topic of research, see e.g. [Gibbs, 1998], [Cho and Saul, 2009], [Wilson and Adams, 2013]. The Gaussian process uses infinitely many fixed basis functions, and typically work as smoothing devices, as opposed to the neural network which use a finite number of adaptive basis functions. Combining the non-parametric flexibility of the GPs with the adaptiveness of neural networks can therefore improve the generalisation capabilities of the GPs, see e.g. [Hinton and Salakhutdinov, 2008], [Wilson et al., 2011], [Bradshaw et al., 2017]. In particular, the authors in [Wilson et al., 2016] used a deep neural network to transform inputs into complex kernels functions, in which the DNN parameters become kernel hyperparameters and [Calandra et al., 2016] used a neural network to transform the input space into a feature space and training a Gaussian process on this space. In [van der Wilk et al., 2017] the authors use a convolutional structure in combination with Gaussian processes to create a convolutional kernel and improve the generalisation abilities; the difference with our work here being that we study the ability of the CNN to by construction perform as a GP. The convolutional structure also has similarities with the additive Gaussian processes [Duvenaud et al., 2011], [Durrande et al., 2012]. These deep and additive kernels bear similarities with the kernel of the Gaussian process that naturally arises when relating the GP to the CNN. As we will show, it is defined as a sum of random variables, in which the kernel in the final output layer can be recursively defined through the previous layer kernel functions.

The relation between neural networks with one hidden layer and GPs was studied in [Neal, 2012]. In particular cases of the activation function the resulting kernels can be computed analytically, see [Williams, 1997] where analytic expressions for the sigmoidal and Gaussian activations are derived. In [Cho and Saul, 2009] the authors provided analytical expressions for the recursive kernel dependency to mimic the behavior in a deep network when using a rectified linear unit activation function. Similarly, the work of [Hazan and Jaakkola, 2015] studied the behaviour of the GP kernels in deep infinite networks. In [Lee et al., 2018] the authors identify the relation between using these kernels as the covariance function for a GP and predicting with Bayesian deep neural networks. The authors of [Matthews et al., 2018] also study the connection between deep fully connected networks to GPs, in particular focussing on the convergence of the deep finite networks to Gaussian processes and the discrepancy between their outputs. Also worth mentioning is the area of deep Gaussian processes [Damianou and Lawrence, 2013] which essentially stack several GPs giving rise to a richer class of probability models.

2 Background

Gaussian processes gained popularity in the machine learning community after the work of [Neal, 2012] showed that the neural network prior tends to a Gaussian process as the number of hidden units tends to infinity. In this section we shortly repeat the main conclusions.

2.1 Gaussian processes

Consider the n-dimensional vector of function values $(f(x^1), ..., f(x^n))$ evaluated at $x^i \in \mathcal{X}$ where $\mathcal{X}$ is the input space, $i = 1, ..., n$. Then for $f : \mathcal{X} \to \mathbb{R}$ we say that f is a Gaussian process [Rasmussen and Williams, 2006], i.e.

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')),$$

if any finite subset $(f(x^1), ..., f(x^n))$ has a multivariate Gaussian distribution. The Gaussian process is defined over the index set $\mathcal{X}$ here equivalent to the input domain, and it is completely specified by its mean function and covariance function as

$$\begin{aligned} m(x) &= \mathbb{E}(f(x)), \\ k(x, x') &= \text{cov}(f(x), f(x')) \\ &= \mathbb{E}((f(x) - m(x))f(x') - m(x'))), \end{aligned}$$

for $x, x' \in \mathcal{X}$. The covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defines the nearness or similarity between two inputs x and x'. Given a sample of input points $X = (x^1, ..., x^n)$, the covariance matrix for this sample $K(X, X) \in \mathbb{R}^{n\times n}$ has entries $K_{i,j} = k(x^i, x^j)$. Consider now the set of training points $\mathcal{D} = \{(x^i, y^i)_{i=1}^N\}$. Let $y = f(x)+\epsilon$, with a Gaussian likelihood, i.e. $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ and assume a Gaussian prior on the function $f \sim \mathcal{GP}(0, k)$. If the prior on f is a GP and the likelihood is Gaussian the posterior on f is also a GP and the predictive distribution is given by

$$\begin{aligned}
p(y_*|x_*, \mathcal{D}) &= \int p(y_*|x_*, f, \mathcal{D})p(f|\mathcal{D})df \\
&= \mathcal{N}(\bar{f}_*, \text{cov}(f_*)), \\
\bar{f}_* &= K(x_*, x)(K(x, x) + \sigma_\epsilon^2 I)^{-1} y, \\
\text{cov}(f_*) &= K(x_*, x_*) \\
&\quad - K(x_*, x)(K(x, x) + \sigma_\epsilon^2 I)^{-1} K(x, x_*).
\end{aligned}$$

2.2 Neural networks and Gaussian processes

Consider an input $x \in \mathcal{X}$ and a feedforward neural network consisting of $L \geq 2$ layers with M hidden nodes in each layer $l = 1, ..., L$. Each layer l in the network then computes for each $i = 1, ..., M$

$$a_i^l(x) = \sum_{j=1}^{M} w_{i,j}^l z_j^{l-1} + b_j^l, \quad z_i^l(x) = h(a_i^l(x))),$$

where we explicitly denote the dependence on a particular input sample x, and $w^l \in \mathbb{R}^{M\times M}$, $b^l \in \mathbb{R}^M$ and $h(\cdot)$ is the nonlinear activation function. Note that in the first layer we compute the linear combination using the input, i.e. $z^0 = x$ and $M = d$. From [Neal, 2012], we know that a neural network with the number of hidden units tending to infinity approaches a Gaussian process prior. Let the weights w^l and biases b^l have zero mean Gaussian distributions with variances $\sigma_w^2 = \nu_w^2/M$ and σ_b^2, respectively. Since a_i^l is a sum of i.i.d. terms – the weights and hidden units – from the CLT it follows that in the limit of $M \to \infty$, the a_i^l will be normally distributed. Similarly consider the joint distribution of $(a_i^l(x^1), ..., a_i^l(x^N))$, where $(x^1, ..., x^N)$ is a finite set of input samples. Since each unit consists of a sum i.i.d. terms with zero mean and fixed covariance, we can apply the multivariate CLT. This tells us that this distribution is multivariate Gaussian, so that

$$\begin{aligned}
a_i^l &\sim \mathcal{GP}(0, k^l(x^p, x^q)), \\
k^l(x^p, x^q) &:= \mathbb{E}\left[a_i^l(x^p) a_i^l(x^q)\right] \\
&= \sigma_b^2 + \sum_{j=1}^{M} \sigma_w^2 \mathbb{E}\left[z_j^{l-1}(x^p) z_j^{l-1}(x^q)\right] \\
&= \sigma_b^2 + \nu_w^2 C^l(x^p, x^q),
\end{aligned}$$

with $C^l(x^p, x^q) = \mathbb{E}\left[h(a_j^{l-1}(x^p))h(a_j^{l-1}(x^q))\right]$ the same for all j. Ways of analytically computing this kernel are discussed in [Williams, 1997], while the authors in [Lee et al., 2018] present a numerical way of computing the kernel for deep neural networks.

3 Convolutional neural networks and Gaussian processes

Consider a convolutional neural network in a one-dimensional setting for ease of notation. Suppose we are given a sequence of inputs $X = (x_1, ..., x_d)$, where each $x_i \in \mathbb{R}$ is assumed to be one-dimensional and a set of outputs $Y = (y_1, ..., y_{d'})$. The output from the first layer is given by convolving the filter w^1 with finite support with the input:

$$a_i^1 = (w^1 * x)(i) = \sum_{j=1}^{M} w_j^l x_{i-j}, \quad z^1 = h(a^1), \quad (1)$$

where $w^1 \in \mathbb{R}^{1\times M}$ and $a^1 \in \mathbb{R}^{1\times N-M+1}$ and the linear combination is passed through the non-linearity $h(\cdot)$ to give $z^1 = h(a^1)$. In each subsequent layer $l = 2, ..., L$ the input feature map, $z^{l-1} \in \mathbb{R}^{1\times N_{l-1}}$, where $1 \times N_{l-1}$ is the size of the output filter map from the previous convolution with $N_{l-1} = N_{l-2} - M + 1$, is convolved with a filter $w^l \in \mathbb{R}^{1\times M}$, to create a feature map $a^l \in \mathbb{R}^{1\times N_l}$:

$$a_i^l = (w^l * z^{l-1})(i) = \sum_{j=1}^{M} w_j^l z_{i-j}^{l-1}, \quad z_i^l = h(a_i^l), \quad (2)$$

with the forecasted output given by $\hat{y} = a^L$. Define the receptive field to be the number of inputs which modify the output. The filter size parameter M thus controls the receptive field of each output node. Without zero padding, in every layer the convolution output has width $N_l = N_{l-1} - M + 1$ for $l = 1, .., L$ and by padding the input with a vector of zeros the size of the receptive field we can control the output size to have the same size as the input. Since all the elements in the feature map share the same weights this allows for features to be detected in a time-invariant manner, while at the same time it reduces the number of trainable parameters.

3.1 Central limit theorem

Consider now the convolutional neural network given in (1) - (2). Let the inputs $(x_1, ..., x_T)$ be the input vector. Initialize all weights in the convolutional neural network to be independent samples from a Gaussian prior $\mathcal{N}(0, \sigma_w^2)$. Consider the output of the convolution in the first layer $a^1 \in \mathbb{R}^d$ where we let d be the placeholder for the dimension of the output. We have

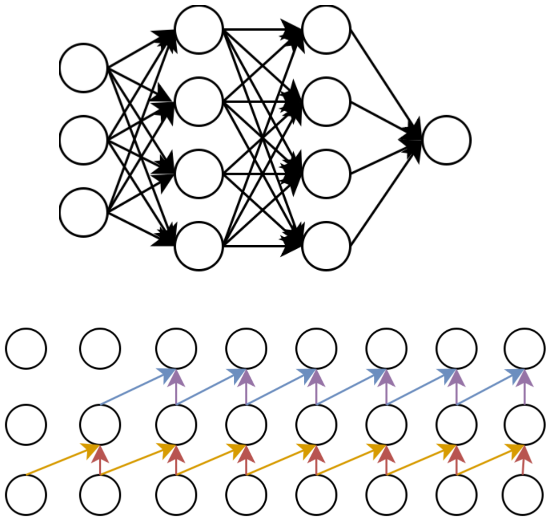

Figure 1: A configuration of a two-layer feedforward network (T) and a convolutional network with a filter size of two, the weights are shared across an input layer as indicated by the similar colors (B).

$a^1 = [a_1^1, a_1^2, ..., a_d^1]^T$ with covariance matrix

$$K^1 = \begin{bmatrix} \mathbb{E}[a_1^1 a_1^1] & \mathbb{E}[a_1^1 a_2^1] & \cdots & \mathbb{E}[a_1^1 a_d^1] \\ \mathbb{E}[a_2^1 a_1^1] & \mathbb{E}[a_2^1 a_2^1] & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{E}[a_d^1 a_1^1] & \cdots & \cdots & \mathbb{E}[a_d^1 a_d^1] \end{bmatrix},$$

in which

$$a_i^1 = \sum_{j=1}^{M} w_j^1 x_{i-j}, \quad K_{i,k}^1 = \sigma_w^2 \sum_{j=1}^{M} \mathbb{E}[x_{i-j} x_{k-j}]. \quad (3)$$

The matrix K^1 thus defines the covariance between the elements of the vector a^1. The terms in the summation for a^1 are independent by the independence between the input and weights and the i.i.d. distribution of the weights

$$\mathbb{E}[w_j^1 x_{i-j} w_k^1 x_{i-k}] = \mathbb{E}[w_j^1 w_k^1]\mathbb{E}[x_{i-j} x_{i-k}] = 0.$$

However, the terms are not necessarily identically distributed. In particular, we have $E[w_j^1 x_{i-j}] = 0$ for all i but $\mathbb{E}[w_j^1 x_{i-j} w_j^1 x_{i-j}] = \sigma_w^2 \mathbb{E}[x_{i-j} x_{i-j}]$ does not have to equal $\sigma_w^2 \mathbb{E}[x_{i-k} x_{i-k}]$. In vector notation

$$V_1^1 = \begin{bmatrix} w_1 x_{1-1} \\ w_1 x_{2-1} \\ \cdots \\ w_1 x_{d-1} \end{bmatrix}, \ \ldots \ , \ V_M^1 = \begin{bmatrix} w_M x_{1-M} \\ w_M x_{2-M} \\ \cdots \\ w_M x_{d-M} \end{bmatrix},$$

where each vector V_j^1 has covariance matrix with elements $\Sigma_{i,k} = \sigma_w^2 x_{i-j} x_{k-j}$. Conditionally on the inputs these vectors are independent but not identically distributed. Therefore, a straightforward application of the multivariate central limit theorem, as is done in the feedforward neural network case, does not apply. Let $|x|^2 = x_1^2 + \cdots + x_d^2$. We introduce the following Theorem as proven in [Bentkus, 2005]

Theorem 1 (A Lyapunov-type bound in $\mathbb{R}^d$)**.** *Let $X_1, \cdots, X_M$ be independent random vectors taking values in $\mathbb{R}^d$ such that $\mathbb{E}[X_i] = 0$ for all i. Let $S = X_1 + \cdots + X_M$. Assume that the covariance operator Σ^2 of S is invertible. Let $Z \sim \mathcal{N}(0, \Sigma^2)$, a centered Gaussian with covariance matrix Σ^2. Let $\mathcal{C}$ stand for the class of all convex subsets of $\mathbb{R}^d$. Then*

$$\sup_{A \in \mathcal{C}} \Big| \mathbb{P}(S \in A) - \mathbb{P}(Z \in A) \Big| \quad (4)$$
$$\leq \mathcal{O}(d^{1/4}) \sum_{i=1}^{M} \mathbb{E}\left[\left| \Sigma^{-1} X_i \right|^3 \right].$$

In particular the above Theorem is able to give a rate of convergence between S and Z but does not require the random variables in the sum to be identically distributed, making it well-suited for application in the convolutional neural network. As a special case of the above we have

Theorem 2 (Independent and identically distributed case)**.** *If the random variables $X_1, \cdots, X_M$ are i.i.d. and have identity covariance $\mathbb{I}_d$, then the lower bound specifies to*

$$\sup_{A \in \mathcal{C}} \Big| \mathbb{P}(S \in A) - \mathbb{P}(Z \in A) \Big|$$
$$\leq \mathcal{O}(d^{1/4}) \mathbb{E}\left[\left| X_1 \right|^3 \right] / \sqrt{M}.$$

Note that the assumption $\Sigma^2 = \mathbb{I}_d$ is not restrictive since we can rescale both S and Z by Σ^{-1} to give the identity covariance matrix. In other words, the above Theorems give a bound on the distance between the probability distribution of the (vector-valued) sum of not necessarily identically distributed variables, i.e. the output a^l, $l = 1, 2, ...$ as computed in a convolutional neural network, and the probability distribution of a multivariate Gaussian random variable, i.e. the output of a Gaussian process in which the kernel is given by Σ^2. In order to be able to measure the closeness of the CNN and the GP we thus need to evaluate the right-hand side of equation 4.

3.2 Kernel specification

The output from the first layer, a^1, has a covariance matrix with elements given in (3). For the output and covariance matrix in the next layers we have

$$a_i^l = \sum_{j=1}^{M} w_j^l z_{i-j}^{l-1}, \quad K_{i,k}^l = \sigma_w^2 \sum_{j=1}^{M} \mathbb{E}\left[z_{i-j}^{l-1} z_{k-j}^{l-1} \right] \quad (5)$$

Again while the terms a_i^l are clearly dependent, the terms in the summation are independent due to the independent and identically distributed weights, but not necessarily identically distributed due to the possibility of the varying covariance. The correlation between nodes in layer l is a sum over the correlations in the receptive field of size M in the previous layer. In this way the convolutional network, as opposed to the neural network, 'averages' the covariances. The rectified linear unit (ReLU) is commonly used as non-linearity in between the layers in both classification and forecasting tasks. Assuming that each layer output is close to a multivariate distribution, the expected value in (5) is over the bivariate normal distribution of z_{i-j}^{l-1} and z_{k-j}^{l-1} with mean zero and covariance matrix with elements $K_{i-j,k-j}^{l}$, $K_{k-j,i-j}^{l}$, $K_{i-j,i-j}^{l}$ and $K_{k-j,k-j}^{l}$. For the ReLU non-linearity the expected value can be computed analytically, see [Cho and Saul, 2009]. We thus obtain the recursive connection (similar to the recursive relationship derived in equation (12) and (13) for a feedforward network in [Cho and Saul, 2009])

$$\begin{aligned} K_{i,k}^{l} =& \frac{\sigma_w^2}{2\pi} \sum_{j=1}^{M} \sqrt{K_{i-j,i-j}^{l-1} K_{k-j,k-j}^{l-1}} \\ & \cdot \left(\sin \theta_{i-j,k-j}^{l-1} + (\pi - \theta_{i-j,k-j}^{l-1}) \cos \theta_{i-j,k-j}^{l-1} \right), \\ \theta_{i,k}^{l} =& \cos^{-1} \left(\frac{K_{i,k}^{l}}{\sqrt{K_{i,i}^{l} K_{k,k}^{l}}} \right), \end{aligned} \tag{6}$$

where we again remark on the summation over the receptive field being the difference with the neural network and the starting point K^1 is given in (3).

Illustrating the convolutional kernel Consider the kernel as defined in (6). As mentioned in e.g. [Lee et al., 2018], as the number of layers grows, the recurrent relation in (5) for a feedforward neural network approaches a fixed point so that the covariance function becomes a constant or piecewise constant map. We can illustrate this for a convolutional network by considering the kernel as a function of θ, the angle between two vectors over which the dot-product in the convolution is computed. In the convolutional network the kernel is a sum, or average, over transforms of the kernels in the previous layer. In this way one would expect –due to the averaging structure– a faster convergence to the flat structure where the structure itself depends on the the angles between the vectors over which the shifting dot-product is computed. For the sake of illustration consider $M = 2$ and two distinct angles θ_1 and θ_2. We compute K^2 as a sum over the kernels in the previous layers $K^1(\theta_1)$ and $K^1(\theta_2)$, see Figure 2, from which we conclude that the averaging indeed results in a more flat structure of the kernel. Compared to the feedforward network in which the flattening depends on the depth, in convolutional networks it is additionally amplified by the convolutional structure, i.e. the averaging over the kernels.

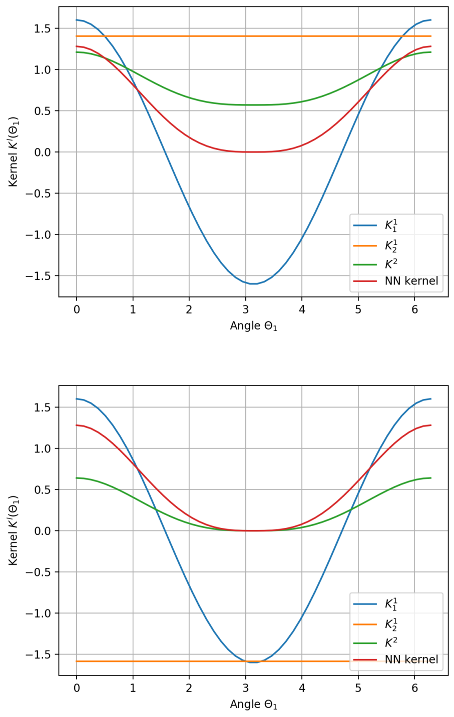

Figure 2: The angular structure of the convolutional kernel as a function of θ_1 with $\theta_2 = 0.5$ (T) and $\theta_2 = 3$ (B) and its evolution with depth for $\sigma_w^2 = \frac{1.6}{M}$. The red line shows the feedforward kernel (without averaging).

4 Gaussian process convergence

In this section we theoretically and numerically study and evaluate the conditions under which the behavior of the CNN output tends to that of a GP. In particular we study the discrepancy between the finite deep CNN and the corresponding Gaussian process in which we use the recurrent relationship for the kernel given in (6), through several intuitive examples. While our focus here is mostly on time series data and forecasting, the overall conclusions generalize to all prediction tasks.

4.1 A theoretical bound

In order to get an idea of the discrepancy between the CNN to the GP, one needs to quantify the term given in the right-hand side of equation 4,

$$\sum_{j=1}^{M} \mathbb{E}\left[|(K^l)^{-\frac{1}{2}} V_j^l|^3 \right],$$

where $V_j^l = [w_j^l z_{1-j}^{l-1}, w_j^l z_{2-j}^{l-1}, \cdots, w_j^l z_{d-j}^{l-1}]^T$ and K^l is defined in (5). This term determines the distance between the CNN output and the GP output. In the case of independent inputs and linear activations we are able to prove the following Lemma on the rate of convergence of the CNN to the GP:

Lemma 3. *Let x_i for $i = 1, ..., T$ be the input variables. Assume $w_j^l \overset{iid}{\sim} \mathcal{N}(0, \sigma_w^2)$. Assume that K^l, the covariance matrix in layer l, is invertible. In the case of the x_i being independent and bounded variables with a finite second moment and when using a linear activation function we have the following bound for some constant $C \in \mathbb{R}$*

$$\sum_{j=1}^{M} \mathbb{E}\left[|(K^l)^{-\frac{1}{2}} V_j|^3\right] \leq dC(M^l)^{-\frac{1}{2}},$$

Proof. By definition

$$K_{i,k}^l = (\sigma_w^2)^l \sum_{j_l=1}^{M} \cdots \sum_{j_1=1}^{M} \mathbb{E}\left[x_{i-j_1-\ldots-j_l} x_{k-j_1-\ldots-j_l}\right].$$

Denote by some constant $\sigma_{i,j}^2$ the variance of x_{i-j}, so that the above simplifies to $K_{i,i}^l = \sum_{j=1}^{M^l} \sigma_{i,j}^2 (\sigma_w^2)^l$, with K^l a diagonal matrix and we change the variables of the summations to an index j here left unspecified. Using the assumption of finite second moments variances we can find for some constant C

$$|(K^l)^{-\frac{1}{2}} V_j|^3 \leq d(CM^l)^{-\frac{3}{2}} |\sigma_w^{-l} V_j^l|^3.$$

We have $\mathbb{E}[|\sigma_w^{-1} V_j^l|^3] \leq \mathbb{E}[|z_{\cdot-j}^{l-1}|^3] \mathbb{E}[||\sigma_w^{-1} w_j^l|^3]$. Using the fact that the squared norm of the variable $\sigma_w^{-1} w_j^l$ follows a chi-squared distribution with 1 degree of freedom we find $\mathbb{E}[|\sigma_w^{-1} w_j^l|^3] = \mathcal{O}(1)$. If the activation function does not increase the norm of the input then $|z_j^l|^3 \leq |a_j^l|^3$ almost surely and in the linear case clearly $|z_j^l|^3 = |a_j^l|^3$. Iterating the above reasoning we then find

$$\mathbb{E}[|l\sigma_w^{-1} V_j^l|^3] \leq \sum_{j=1}^{M^l} |x_{\cdot-j}|^3,$$

which using the boundedness of x_i concludes the proof. □

In the non-linear case, it is more complex to quantify the convergence with the number of layers. Due to the compositional nature of the CNN, increasing the number of layers causes the receptive field of the network to grow. In particular when using a rectified linear unit activation which simply maps the input to itself or to zero, the output in the l-th layer can be seen as a sum of a certain number of inputs x_i. However, due to the non-linearity it is hard to quantify how many and which inputs are used and therefore computing the kernel in the GP in a correct way so as to obtain a close enough correspondence to a CNN is not trivial. In the linear, independent case we obtain a rate of convergence of order $\sqrt{M^l}$, so one could postulate that also in the non-linear, dependent case the convergence rate should be dependent on the *total* receptive field in this way obtaining a closer correspondence with the GP as the number of layers grows. However, the non-linearities applied in each layer change the underlying dependency structure in such a way that it no longer becomes possible to propagate the dependency on the inputs through the layers. Application of the non-linear activation seems to allow one to diverge from GP behavior in the output: even if the receptive field is growing, the convergence to the GP is only governed by the filter size, and not the total receptive field, as we shall also show in the numerical section. This provides a probabilistic justification for using the non-linear activations: besides being able to transform the input data in non-linear ways, it also allows one to diverge from Gaussian output behavior and hopefully result in more complex probability distributions.

The rate of convergence between the CNN output and the GP thus depends on the quantification of the term in (4). In the case of sufficiently independent inputs, as is e.g. a common assumption when working with financial returns, we showed that the discrepancy between the CNN and GP outputs to converges as $M \to \infty$. In the non-independent case, for a linear activation, we have

$$K^l = (\sigma_w^2)^l \sum_{j=1}^{M^l} \Sigma^{(j)}, \text{ where } \Sigma_{i,k}^{(j)} = x_{i-j} x_{k-j},$$

and the convergence will depend on the inverse of the square root of this matrix, or in other words on properties of the distribution of the input variables themselves. The empirical properties of the data can be used in order to understand whether GP behavior will be present in the network.

4.2 Numerical discrepancy

The empirical discrepancy is determined using the maximum mean discrepancy (MMD) introduced in [Gretton et al., 2012] and applied to measuring the similarity between GPs and feedforward neural networks in [Matthews et al., 2018]. The MMD between two distributions $\mathcal{P}$ and $\mathcal{Q}$ is defined as

$$\mathcal{MMD}(\mathcal{P}, \mathcal{Q}, \mathcal{H}) := \sup_{||h||_{\mathcal{H}} \leq 1} \left[\mathbb{E}^{\mathcal{P}}[h] - \mathbb{E}^{\mathcal{Q}}[h]\right],$$

for which we use the unbiased estimator of squared MMD given in equation (3) in [Gretton et al., 2012]. For the empirical computations we take an input X, pass this through 500 convolutional neural networks and draw 500 samples from the GP with the covariance kernel corresponding to the depth of the CNN as defined in (6), and compare the results for different network parameters using the unbiased MMD estimator. We plot for comparison also the MMD estimator between two Gaussian processes with RBF kernels with length scales $l = \sqrt{2}$ and $l = 4\sqrt{2}$.

Prior discrepancy Let $x_i \overset{iid}{\sim} \mathcal{N}(0,1)$ for $i = 1, ..., d$ with $d = 50$ and let $\sigma_w^2 = 1$. When computing the covariance matrix we condition on the input so that we have $K_{i,k}^1 = \sigma_w^2 \sum_{j=1}^{M} x_{i-j} x_{k-j}$ so that the terms in the convolutional sum are not necessarily identically distributed; we compute the covariance function using (6) and using a Monte Carlo (MC) evaluation of the expected value in (5) by using 1000 Monte Carlo simulations in which we sample the weights $w_i \overset{iid}{\sim} \mathcal{N}(0, \sigma_w^2)$. Figure 3 shows the results of the discrepancy between the GP and CNN. In the linear case we observe a fast convergence regardless of the number of layers, as is in accordance with the results of Lemma 3. In the non-linear case we observe a slower convergence in M as we increase l, but nevertheless GP behavior is reached. The slower convergence can be explained by the application of the non-linearities which essentially violate the dependency structure and no longer allow for the output in the network to depend on the total receptive field but just on the number of inputs from the previous layer. We furthermore note that the expression for the kernel in (6) is computed under the assumption of a sufficiently normal output in each layer, which in the GP, similar to the feedforward network, is only attained in the limit. Nevertheless, compared to the MC computation of the kernel, the discrepancy between the two in the output prior seems to be sufficiently small, so that the analytical expression in (6) is also able to mimic the recurrent relation in the deep convolutional network. When using CNNs for sequence forecasting a common setup is using a filter of size 2-8 and 2-5 layers depending on the required receptive field, see e.g. [Bai et al., 2018], [Borovykh et al., 2017]. From Figure 3 one can conclude that the CNN indeed tends to behave like a Gaussian process, nevertheless this behavior can be more or less avoided for a large number of layers and a relatively small filter size.

Posterior discrepancy Consider now x_i to be an autoregressive time series of order three for $i = 1, ..., d$, a time series on which the CNN is well-able to learn the corresponding dynamics, let $\sigma_w^2 = 1$ and compute the covariance conditional on the input. In Figure **??** we plot the GP and CNN prior discrepancy and see that using the non-linear activation results in a slower convergence when using more layers, nevertheless GP behavior is still obtained. Figure 4 shows the posterior GP mean and variance as well as the mean and variance of the CNN output obtained by training $N = 100$ CNNs with random initialization (e.g. a simplified form of uncertainty estimation with an ensemble [Lakshminarayanan et al., 2016]; the network is trained by minimizing the mean squared error which is not a completely valid metric for capturing the uncertainty, nevertheless in our simplified example is enough to at least show the discrepancy between the posteriors). As expected, we conclude that if the priors of a GP and CNN are corresponding also their posteriors will be close.

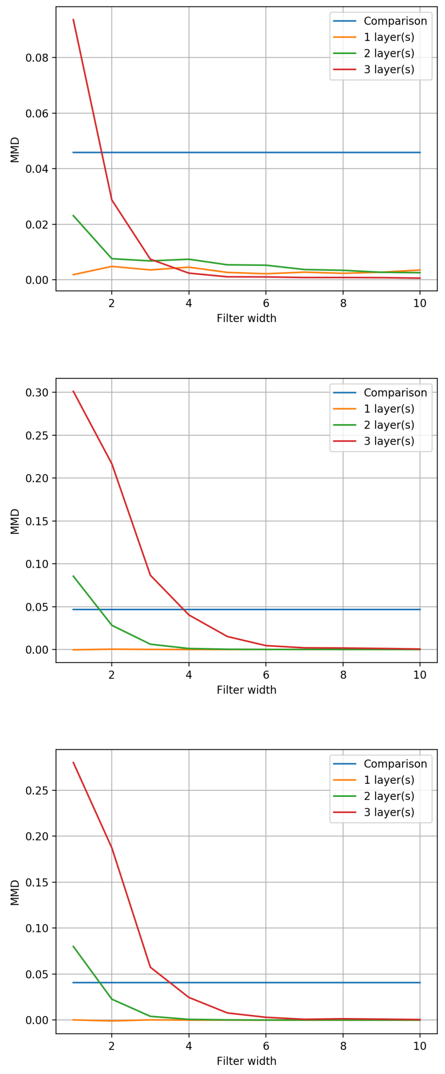

Figure 3: MMD between the GP and CNN with a linear activation (T), a ReLU activation using (6) (C) and a ReLU using the MC evaluation of (5) (B) for not identically distributed inputs.

4.3 Extension to CNNs for image processing

The convolutional network we considered before is applied to a one-dimensional input. Typically, in image processing the convolutions are performed across a multi-dimensional input. Consider the output of a layer l in which C_l filters were applied to an input of with C_{l-1}

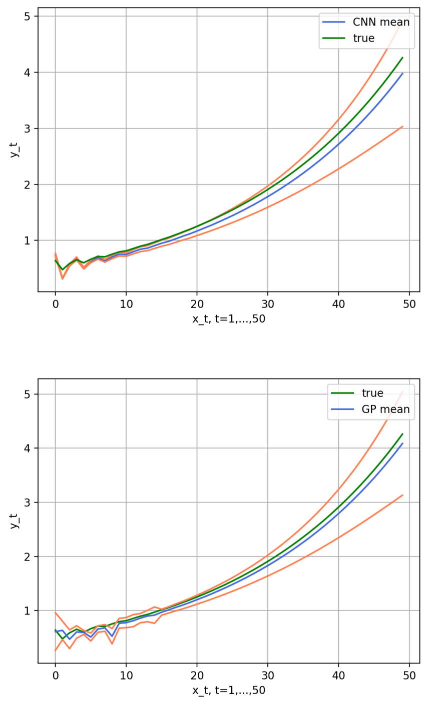

Figure 4: A comparison between the trained CNN mean (T) and posterior inference in the corresponding Gaussian process (B) with credible confidence intervals. The CNN has one hidden layer and a filterwidth of three.

channels,

$$a^l_{i,k,c} = \sum_{j,h=1}^{M} \sum_{c=1}^{C_{l-1}} w^l_{j,h,c} z^{l-1}_{i-j,k-h,c}, \text{ for } c = 1, ..., M_l,$$

with $x \in \mathbb{R}^{H\times N\times C_{l-1}}$ and $w \in \mathbb{R}^{M\times M\times C_{l-1}}$, where $H \times N$ is the height and width of the input, and $M \times M$ the filter size. The weights are as usual initialized independently from a Gaussian prior. The CNN thus computes a summation over $M \times M \times C_{l-1}$ terms which are independent (due to the weights being independent) but not necessarily identically distributed (note: the outputs for the different filters are identically distributed, each filter is applied to the same input, in this way the filters are related to the hidden nodes in the feedforward network). Therefore, the Lyapunov bound still holds, and we note here that the convergence to a GP will hold even more strongly, as we sum over more terms. As an example, consider Figure 3 where we see that GP behavior is obtained for three layers using a one-dimensional filterwidth of 10; for images this would mean that using a filter of size 4×4 would certainly result in GP behavior.

5 Conclusion and discussion

In this paper we studied the convolutional neural networks by considering the deep network from a Gaussian process perspective. We extended the state of knowledge about GPs and deep networks to the convolutional network. In particular, using a Lyapunov-type bound for the case in which the inputs are not identically distributed we are able to obtain a CLT-like result on the discrepancy between the CNN and GP. In particular the Lyapunov-bound defines the rate of convergence between the CNN and GP output through the covariance kernel, allowing one to study the discrepancy and convergence rate through the kernel. We proved a theoretical bound for the case in which the inputs into the CNN are independent and a linear activation function is used and showed that the discrepancy is influenced by both the filter width and the number of layers. In the case of the inputs being dependent the bound will clearly depend also on the attributes of the data itself. The empirical tests show that using both linear and non-linear activations results in situations in which the CNN tends to behave like a GP, even if for the non-linear activations the convergence is slower, which coincides with the results obtained in [Matthews et al., 2018] for the feedforward network. From a Bayesian perspective, if the priors of the two are indeed close enough we can use the analytical properties of GPs to obtain the uncertainty estimates in the CNN, thereby avoiding computationally intensive options such as Markov Chain Monte Carlo methods.

The downside of the expression in (6) is that it assumes that the outputs in *each* layer are multivariate normally distributed. In the convolutional neural network one interesting difference with the neural network is the compositionality of the network, so that the receptive field, or the number of inputs that influences the output, grows with the number of layers and the filter width. In the case of linear and independent inputs we proved that the convergence rate is then a function of the total receptive field, in other words the compositionality allows for a faster convergence. In the non-linear setting however the receptive field does not seem to play a role in the convergence to the GP, since as obtained in the numerical examples, a larger number of layers results in a larger discrepancy with the GP. This coincides with the results obtained in [Matthews et al., 2018] in which the authors conclude that the rate at which the limit distribution is obtained in a feedforward neural network affects the distribution in the subsequent layers. Furthermore we postulate that also the non-linearity plays a role in this, changing the distribution of the outputs in such a way that the compositionality of the CNN is no longer propagated through the network. Nevertheless, a question remaining for further research is proving this convergence rate in the non-linear setting rigorously in the convolutional network, in order to obtain complete results on the effects of the total receptive field of the network on the GP behavior.

Concluding, casting the network into a known framework such as the GPs provides interesting insights into the workings of the CNN, such as its distribution and the properties (a small filter size, many layers) that can allow its output distribution to deviate from a Gaussian and learn other, more complex distributions. This work was to the best of our knowledge the first attempt at doing so for the convolutional network structure, which possesses a very different structure compared to the feed-forward neural network. In this way for example the recently obtained results on how to sample from the posterior distribution using stochastic gradient descent [Mandt et al., 2017] might differ for the two networks. We hope that illustrating the similarities between CNNs and particular GPs our results will further research in both of these areas.

References

[Bai et al., 2018] Bai, S., Kolter, J. Z., and Koltun, V. (2018). Convolutional sequence modeling revisited. *ICLR Workshop Track*.

[Bentkus, 2005] Bentkus, V. (2005). A Lyapunov-type bound in $\mathbb{R}^d$. *Theory of Probability & Its Applications*, 49(2):311–323.

[Borovykh et al., 2017] Borovykh, A., Bohte, S., and Oosterlee, C. W. (2017). Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*.

[Bradshaw et al., 2017] Bradshaw, J., Matthews, A. G. d. G., and Ghahramani, Z. (2017). Adverial examples, uncertainty and transfer testing robustness in gaussian process hybrid deep networks. *ArXiv*.

[Calandra et al., 2016] Calandra, R., Peters, J., Rasmussen, C. E., and Deisenroth, M. P. (2016). Manifold gaussian processes for regression. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 3338–3345. IEEE.

[Cho and Saul, 2009] Cho, Y. and Saul, L. K. (2009). Kernel methods for deep learning. In *Advances in neural information processing systems*, pages 342–350.

[Damianou and Lawrence, 2013] Damianou, A. C. and Lawrence, N. D. (2013). Deep gaussian processes. *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[Durrande et al., 2012] Durrande, N., Ginsbourger, D., and Roustant, O. (2012). Additive covariance kernels for high-dimensional Gaussian process modeling. *Annales de la Faculté de Sciences de Toulouse*, 21:481.

[Duvenaud et al., 2011] Duvenaud, D. K., Nickisch, H., and Rasmussen, C. E. (2011). Additive Gaussian processes. In *Advances in neural information processing systems*, pages 226–234.

[Gibbs, 1998] Gibbs, M. N. (1998). *Bayesian Gaussian processes for regression and classification*. PhD thesis, University of Cambridge Cambridge, England.

[Gretton et al., 2012] Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773.

[Hazan and Jaakkola, 2015] Hazan, T. and Jaakkola, T. (2015). Steps toward deep kernel methods from infinite neural networks. *arXiv preprint arXiv:1508.05133*.

[Hinton and Salakhutdinov, 2008] Hinton, G. E. and Salakhutdinov, R. R. (2008). Using deep belief nets to learn covariance kernels for gaussian processes. In *Advances in neural information processing systems*, pages 1249–1256.

[Karpathy et al., 2014] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732.

[Kim, 2014] Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 25*, pages 1097–1105.

[Lakshminarayanan et al., 2016] Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2016). Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*.

[Lee et al., 2018] Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. (2018). Deep neural networks as gaussian processes. *Submitted to ICML*.

[Mandt et al., 2017] Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). Stochastic gradient descent as approximate bayesian inference. *arXiv preprint arXiv:1704.04289*.

[Matthews et al., 2018] Matthews, A. G., Hron, J., Rowland, M., Turner, R. E., and Ghahramani, Z. (2018). Gaussian process behaviour in wide deep neural networks. *ICML*.

[Neal, 2012] Neal, R. M. (2012). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

[Rasmussen and Williams, 2006] Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian processes for machine learning*, volume 1. MIT press Cambridge.

[van den Oord et al., 2016] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. *ArXiv e-prints*.

[van der Wilk et al., 2017] van der Wilk, M., Rasmussen, C. E., and Hensman, J. (2017). Convolutional gaussian processes. In *Advances in Neural Information Processing Systems*, pages 2845–2854.

[Williams, 1997] Williams, C. K. (1997). Computing with infinite networks. In *Advances in neural information processing systems*, pages 295–301.

[Wilson and Adams, 2013] Wilson, A. and Adams, R. (2013). Gaussian process kernels for pattern discovery and extrapolation. In *International Conference on Machine Learning*, pages 1067–1075.

[Wilson et al., 2016] Wilson, A. G., Hu, Z., Salakhutdinov, R. R., and Xing, E. P. (2016). Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pages 2586–2594.

[Wilson et al., 2011] Wilson, A. G., Knowles, D. A., and Ghahramani, Z. (2011). Gaussian process regression networks. *arXiv preprint arXiv:1110.4411*.