Short Bio

2009-2015: Bachelor and Master of statistics, minor in computer science. LMU Munich.
2014-2018: Dr. rer. nat. in computer science. LMU Munich, supervised by Prof. Dr. Volker Tresp.

2018-2022: Research Scientist at Siemens AG, Munich.
2022- : Senior Key Expert for robust AI at Siemens AG, Munich.
2019- : Co-lecturer “Deep Learning and AI” at LMU Munich.

SIEMENS

Tensorization and uncertainty quantification in Deep Learning

Dr. Yinchong Yang, Siemens AG

Invited Talk at Cyprus Institute, 2023-05-23

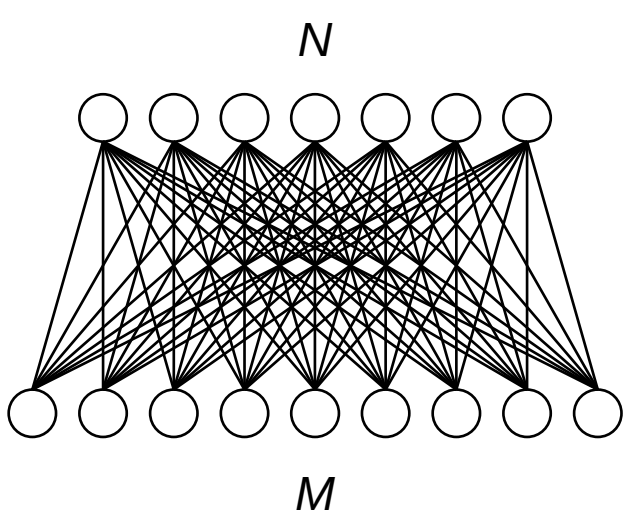
SIEMENS

Tensorization of neural networks in Deep Learning

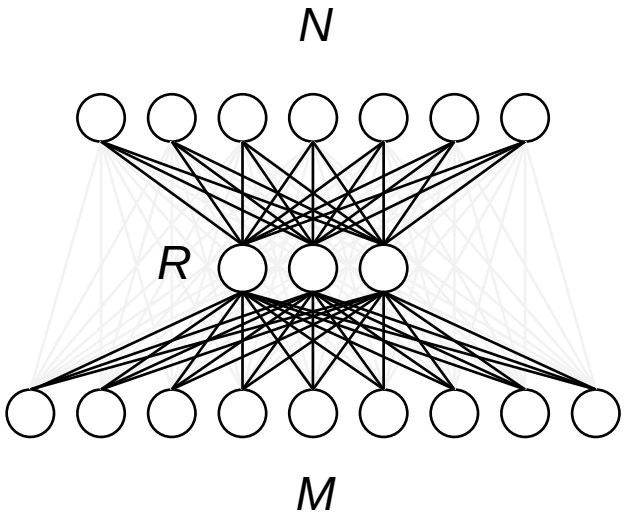
“Representing large weight matrices in NN with product of small tensors…”

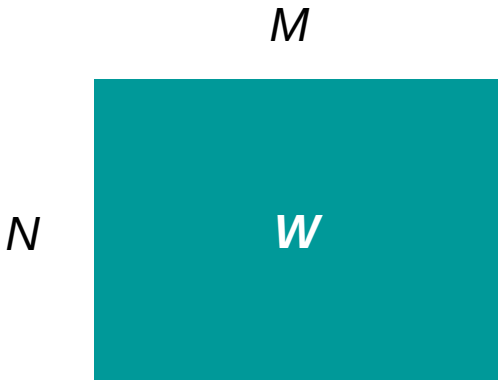
SIEMENS

A motivating example

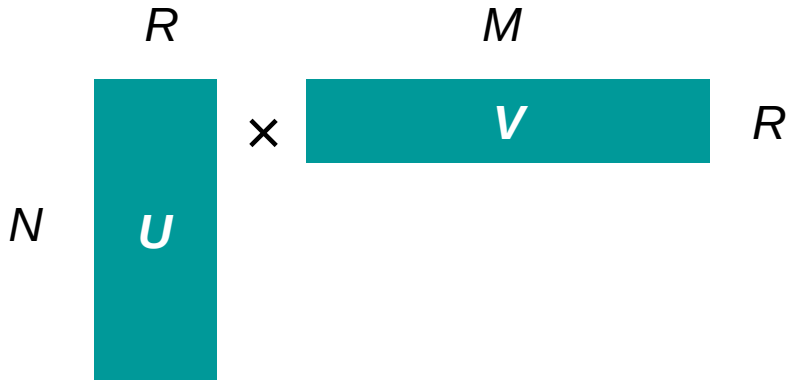


≈





≈

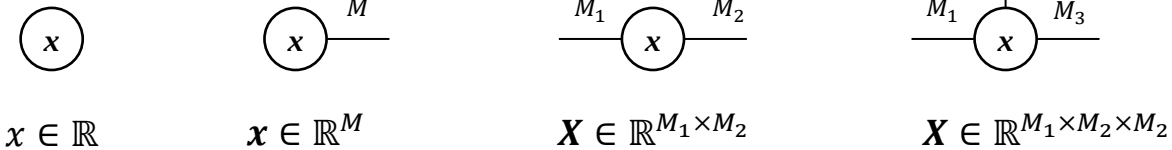


SIEMENS

Preliminary: Tensor Graph and Einstein Summation Notation

In the syntax of tensor graph notation:

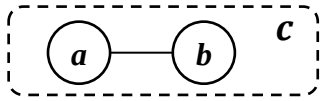
- Node = tensor
- Edge = “mode” / dimensionality

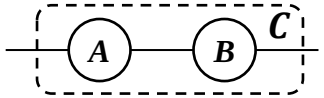


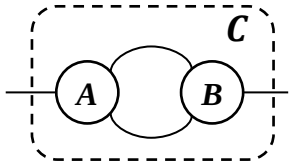
$x \in \mathbb{R}$ $\quad \boldsymbol{x} \in \mathbb{R}^{M}$ $\quad \boldsymbol{X} \in \mathbb{R}^{M_1 \times M_2}$ $\quad \boldsymbol{X} \in \mathbb{R}^{M_1 \times M_2 \times M_2}$

- Connection = tensor contraction
  - Combining multiple tensors to form a new one.
  - A graph representation of Einstein’s summation.

Tensor graph notation	Matrix notation	Einstein summation notation	pytorch

	$c = \langle \boldsymbol{a}, \boldsymbol{b} \rangle$	$c = a(n)b(n)$	`a = torch.randn(5)`
			`b = torch.randn(5)`
			`torch.einsum('n,n->', a, b)`
			`# torch.inner(a, b)`

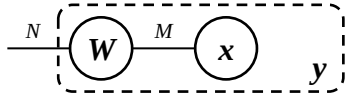
	$\boldsymbol{C} = \boldsymbol{AB}$	$\boldsymbol{C}(m,k) = \boldsymbol{A}(m,n)\boldsymbol{B}(n,k)$	`A = torch.randn(3, 5)`
			`B = torch.randn(5, 4)`
			`torch.einsum('mn,nk->mk', A, B)`
			`# torch.matmul(A, B)`

	?	$\boldsymbol{C}(n,r) = \boldsymbol{A}(m,n,k)\boldsymbol{B}(m,k,r)$	`A = torch.randn(5, 10, 7)`
			`B = torch.randn(5, 7, 2)`
			`torch.einsum('mnk,mkr->nr', A, B)`

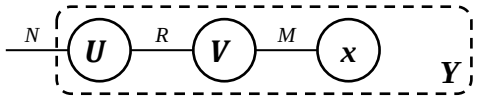
SIEMENS

Tensorized neural networks

- A fully connected layer (omitting the bias) $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$ can be formulated as contraction:
  - $\boldsymbol{y}(n) = \boldsymbol{W}(n,m)\boldsymbol{x}(m)$, with $\boldsymbol{W} \in \mathbb{R}^{N\times M}, \boldsymbol{x} \in \mathbb{R}^{M}$.
  - Size: $N \times M$.

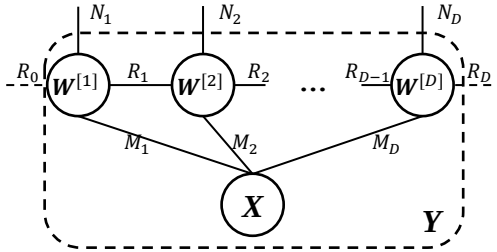


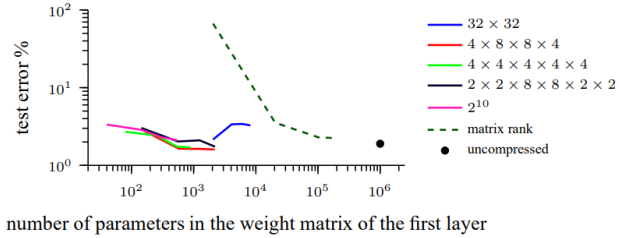
- A bottleneck FC layer as:
  - $\boldsymbol{y}(n) = \boldsymbol{V}(n,r)\boldsymbol{U}(r,m)\boldsymbol{x}(m)$ with $\boldsymbol{V} \in \mathbb{R}^{N\times R}, \boldsymbol{U} \in \mathbb{R}^{R\times M}, \boldsymbol{x} \in \mathbb{R}^{M}$.
  - Size: $R \times (M + N)$.



- A tensor-train layer [Novikov et al. 2015] closely related to Matrix Product State.
  - $\boldsymbol{y}(n_1, n_2, \ldots, n_D) =$
    $\boldsymbol{W}^{[D]}(n_D, r_{D-1}, r_D, m_D) \ldots \boldsymbol{W}^{[2]}(n_2, r_1, r_2, m_2)\, \boldsymbol{W}^{[1]}(n_1, r_0, r_1, m_1)\, \boldsymbol{x}(m_1, m_2, \ldots, m_D),$
  - Size: $\sum_{d=1}^{D} M_d N_d R_{d-1} R_d$ instead of $\mathrm{M} \times N = \prod_{d=1}^{D} M_d N_d$

$$M = M_1 \times M_2 \times \ldots \times M_D$$
$$N = N_1 \times N_2 \times \ldots \times N_D$$



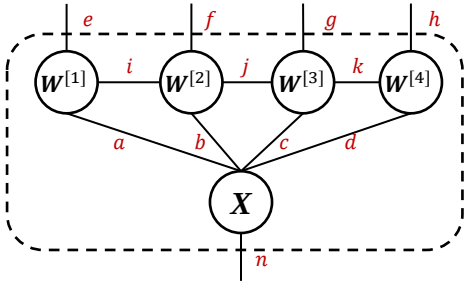


Novikov, Alexander, et al. "Tensorizing neural networks." *NeurIPS* 2015.

SIEMENS

Tensorized neural networks

- Implementation in pytorch:
  - Step 1: decide the decomposition of input and output modes, as well as the rank.
    - Example: 4096 → 16 can be decomposed into $(8 \times 8 \times 8 \times 8) \rightarrow (2 \times 2 \times 2 \times 2)$ with rank 2
  - Step 2: draw a tensor train layer and name the edges.
    - Example:




```
opt = torch.optim.Adam([W1, W2, W3, W4], lr=5e-2)
loss = torch.nn.MSELoss()

for i in range(10000):
    y_hat = torch.einsum('aei, bfij, cgjk, dhk, nabcd -> nefgh', W1, W2, W3, W4, X)
    error = loss(y_hat.reshape(n_samples, -1), y.reshape(n_samples, -1))
    error.backward()
    opt.step()
    opt.zero_grad()
```

  - Step 3: implement an torch.einsum operation with given names in step 2.
  - Step 4: implement the backward pass.

Tensorized neural networks

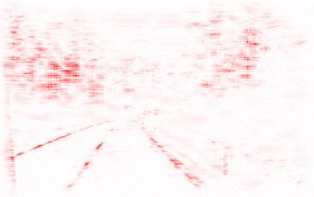
	Accuracy	# Parameters	Runtime
TT-MLP	0.427 ± 0.045	7,680	902s
GRU	0.488 ± 0.033	44,236,800	7,056s
LSTM	0.492 ± 0.026	58,982,400	8,892s
TT-GRU	**0.813 ± 0.011**	3,232	1,872s
TT-LSTM	0.796 ± 0.035	3,360	2,160s

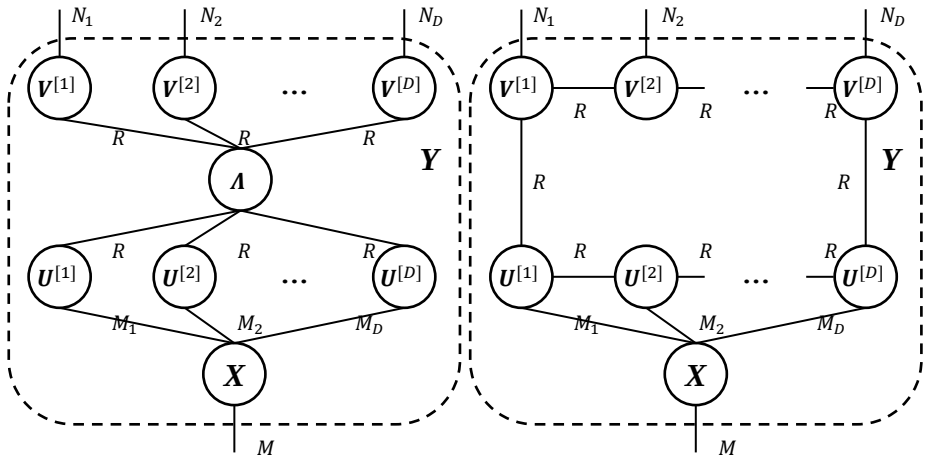
Original: (Liu et al., 2009)	0.712
(Liu et al., 2013)	0.761
(Hasan & Roy-Chowdhury, 2014)	0.690
(Sharma et al., 2015)	**0.850**
Our best model (TT-GRU)	0.813

- Line of research: new architectures based on tensorized neural network:
  - Tensorized CNN [Garipov et al. 2016],
  - Tensorized RNN [Yang et al. 2017, Tjandra et al. 2017],

- Line of research: new applications:
  - Time series data [Yu et al. 2017]
  - Video data – e.g. autonomous driving (contribution by Max Pittner) with LRP.
  - Sparse tabular data [Yang et al. 2017]

- Line of research: other decompositions
  - CP and Tucker [Pan et al. 2023] (left)
  - Tensor-Ring [Wang et al. 2018] (right)



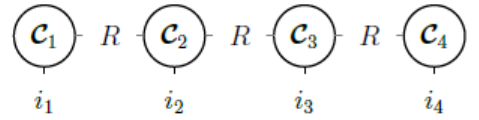


Garipov, Timur, et al. "Ultimate tensorization: compressing convolutional and fc layers alike." *NeurIPS* (2016) *Workshop*.
Tjandra, Andros, Sakriani Sakti, and Satoshi Nakamura. "Compressing recurrent neural network with tensor train." *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017.
Yu, Rose, et al. "Long-term forecasting using tensor-train rnns." *Arxiv* (2017).
Yang, et al. "Modeling progression free survival in breast cancer with tensorized recurrent neural networks and accelerated failure time models." MLHC, PMLR, 2017.
Pan et al. https://tednet.readthedocs.io/en/latest/ visited on 2023-05-18
Wang, Wenqi, et al. "Wide compression: Tensor ring nets." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

SIEMENS

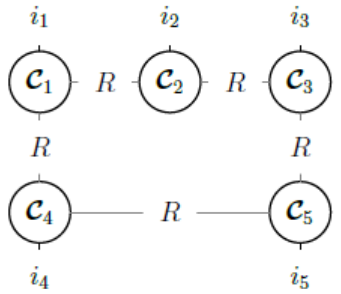
Tensorized neural networks

$$\boldsymbol{A}_{TT} = \begin{pmatrix} 0 & R & 0 & 0 \\ R & 0 & R & 0 \\ 0 & R & 0 & R \\ 0 & 0 & R & 0 \end{pmatrix} \equiv$$

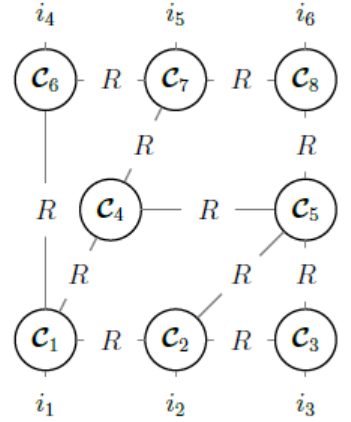


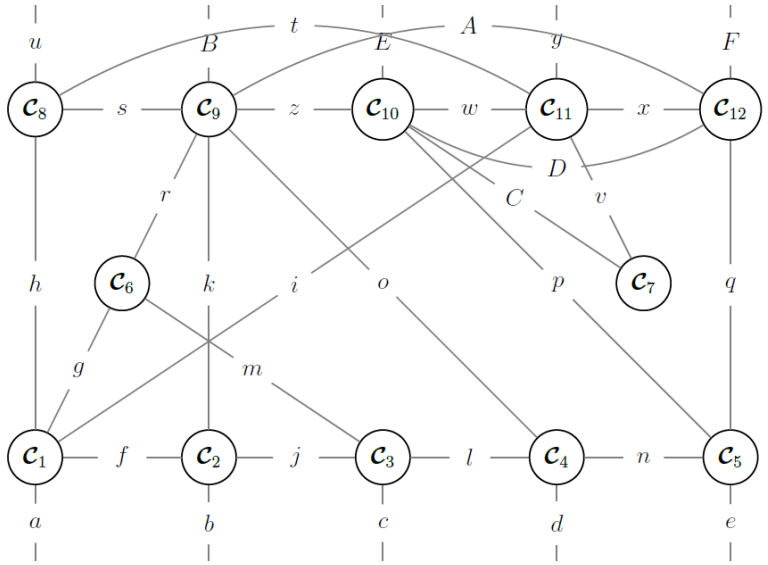
- Flexible topology [Li and Sun 2020, Master thesis Arber Qoku]
  - Representing the core tensor topology with adjacency matrix (vectorized lower-/upper triangle).
  - Optimize the topology of core tensors as a hyper-parameter via evolution strategy and Bayesian optimization:

$$\boldsymbol{A}_{TR} = \begin{pmatrix} 0 & R & 0 & R & 0 \\ R & 0 & R & 0 & 0 \\ 0 & R & 0 & 0 & R \\ R & 0 & 0 & 0 & R \\ 0 & 0 & R & R & 0 \end{pmatrix} \equiv$$



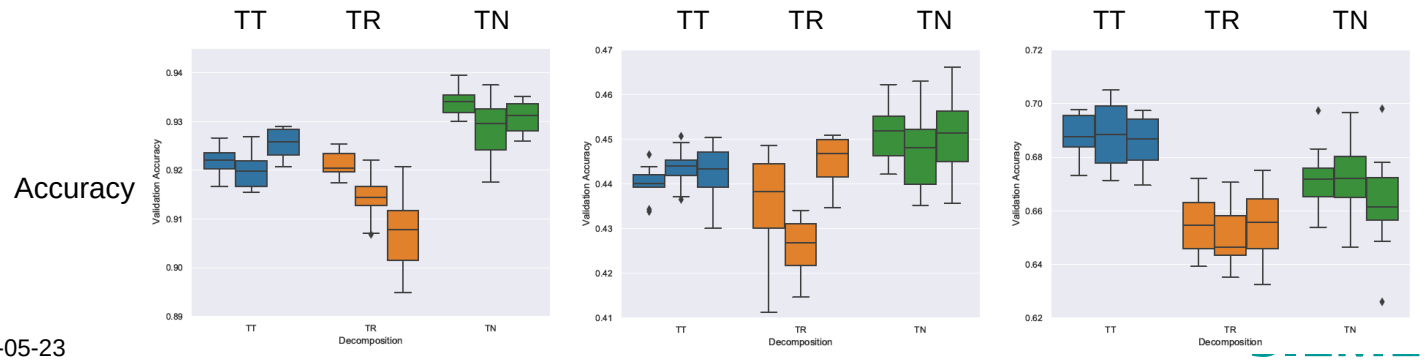
$$\boldsymbol{A}_{TN} = \begin{pmatrix} 0 & R & 0 & R & 0 & R & 0 & 0 \\ R & 0 & R & 0 & R & 0 & 0 & 0 \\ 0 & R & 0 & 0 & R & 0 & 0 & 0 \\ R & 0 & 0 & 0 & R & 0 & R & 0 \\ 0 & R & R & R & 0 & 0 & 0 & R \\ R & 0 & 0 & 0 & 0 & 0 & R & 0 \\ 0 & 0 & 0 & R & 0 & R & 0 & R \\ 0 & 0 & 0 & 0 & R & 0 & R & 0 \end{pmatrix} \equiv$$





Random Search
Bayesian optimization
Evolution strategy

MNIST CIFAR10 Paul15



Li, Chao, and Zhun Sun. Evolutionary topology search for tensor network decomposition. *ICML*, 2020.

Uncertainty quantification with GP in Deep Learning

“Combining the predictive distribution of GP with representation learning by NN.”

SIEMENS

Gaussian Processes Recap

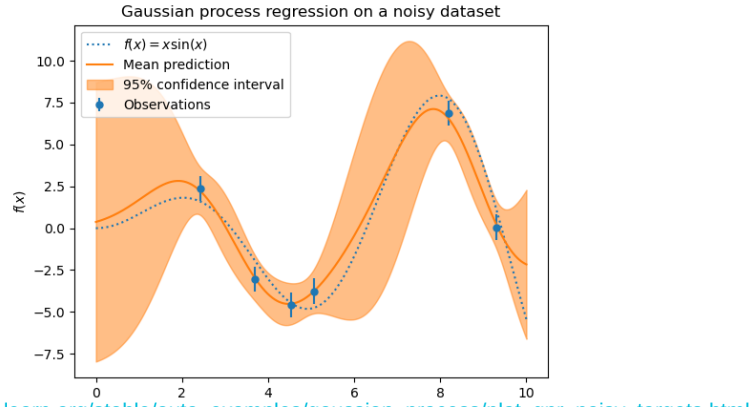
- Standard Gaussian Process regression:
  - $p(y_*|\boldsymbol{x}_*, D) = \int p(y_*|\boldsymbol{x}_*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|D) d\boldsymbol{\theta}$

$$= N(\frac{1}{\sigma^2}\boldsymbol{k}_*^T(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}, k_{**} - \boldsymbol{k}_*^T(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}_* + \sigma^2)$$

with

$$\boldsymbol{k}_* = (k(\boldsymbol{x}_*, \boldsymbol{x}_i))_{i=1}^n \epsilon \mathbb{R}^n \text{ and } \boldsymbol{K} = \left(k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)_{i=1,j=1}^{n,n} \epsilon \mathbb{R}^{n \times n}$$

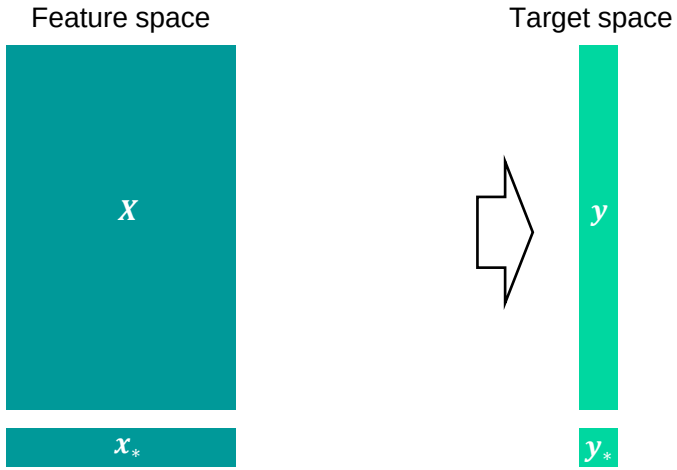
being the kernel function that measures the similarity between data points.



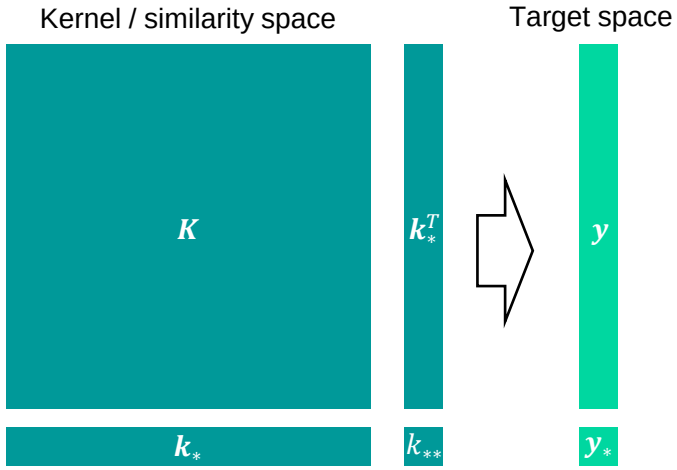
https://scikit-learn.org/stable/auto_examples/gaussian_process/plot_gpr_noisy_targets.html

- The computational complexity is $\boldsymbol{O}(n^3)$ due to the matrix inverse.

Parametric models:



Non-Parametric models:



SIEMENS

Scalable Gaussian Processes

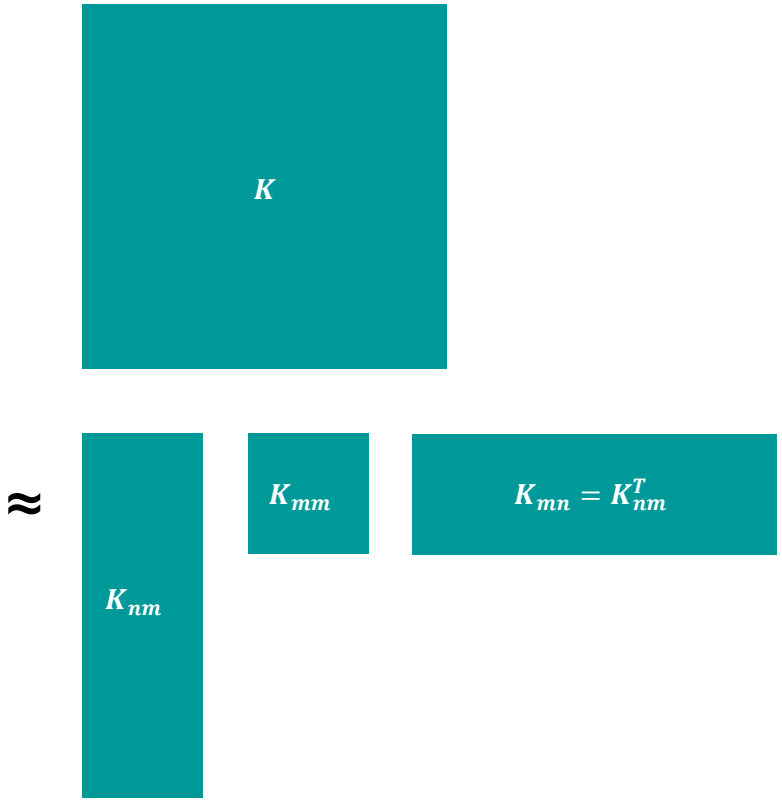
- Various approaches have been proposed to approximate the full covariance matrix. But the idea remains the same: representing the whole training set with a few "*inducing points*":

$$\boldsymbol{K} \approx \boldsymbol{Q} = \boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-1}\boldsymbol{K}_{mn}$$

  so that it can be inverted using the Woodbury formula [Williams & Seeger 2000]. That is,

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) \approx \boldsymbol{k}(\boldsymbol{x}_i, \boldsymbol{Z})^T \boldsymbol{K}_{mm}^{-1} \boldsymbol{k}(\boldsymbol{x}_j, \boldsymbol{Z})$$

- The set of inducing points $\boldsymbol{Z}$ could also be learned as parameter:
  - FITC [Snelson et al. 2000]
  - VFE [Titias 2009]
  - SVGP [Hensman et al. 2013]
  - PPGP [Jankowiak et al. 2020]



Williams and Seeger. Using the Nyström method to speed up kernel machines. *NeurIPs* 2000.
Snelson et al. Sparse Gaussian processes using pseudo-inputs. *NeurIPs*. 2006.
Titsias. Variational learning of inducing variables in sparse Gaussian processes. *AISTATS* 2009.
Hensman et al. Gaussian Processes for Big Data. *UAI.* 2013.
Jankowiak et al. Parametric gaussian process regressors. *ICML*, 2020.

SIEMENS

Scalable Gaussian Processes for collaborative filtering

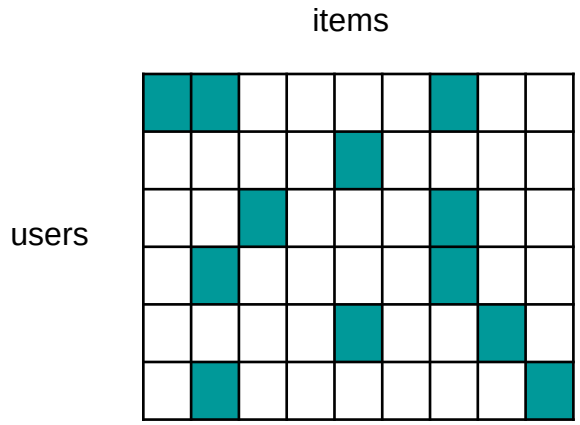
- Collaborative filtering: modeling user-item interaction, applied in recommender systems.
- Classical approaches include matrix decomposition such as SVD and NMF, more advanced methods such as multiway NN [Nickel et al. 2015].

$$y_{i,j} = f(\boldsymbol{a}_i, \boldsymbol{b}_j)$$

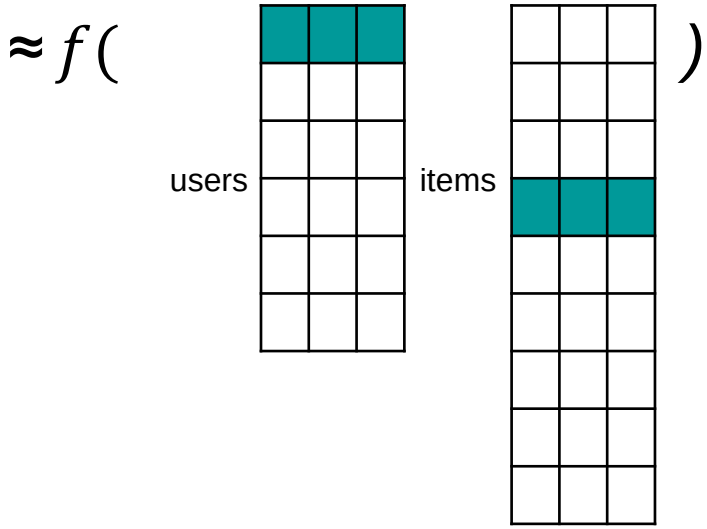
  - with

$$\boldsymbol{Y} \in \mathbb{R}^{I \times J}, \boldsymbol{A} \in \mathbb{R}^{I \times r}, \boldsymbol{B} \in \mathbb{R}^{I \times r}$$

- The expressiveness of the model depends on the choice of function f.



- GP-LVM is one of the very few methods are capable of expressing predictive uncertainty, but doesn't scale well to large and sparse user-item matrices.



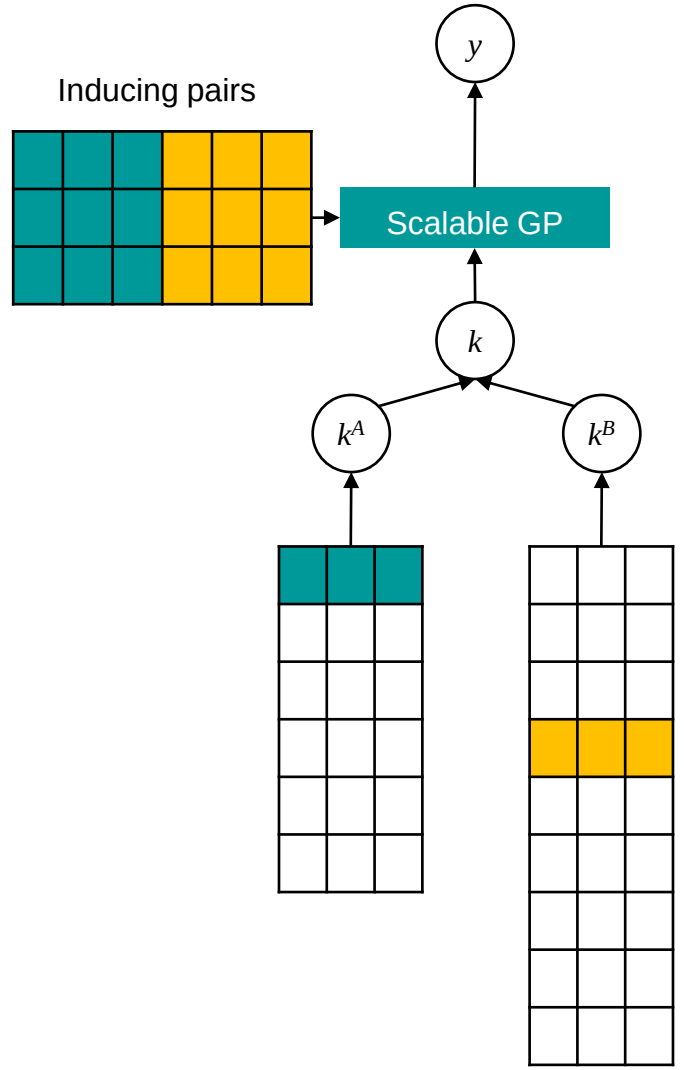
Nickel, Maximilian, et al. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE* 104.1 (2015): 11-33.

SIEMENS

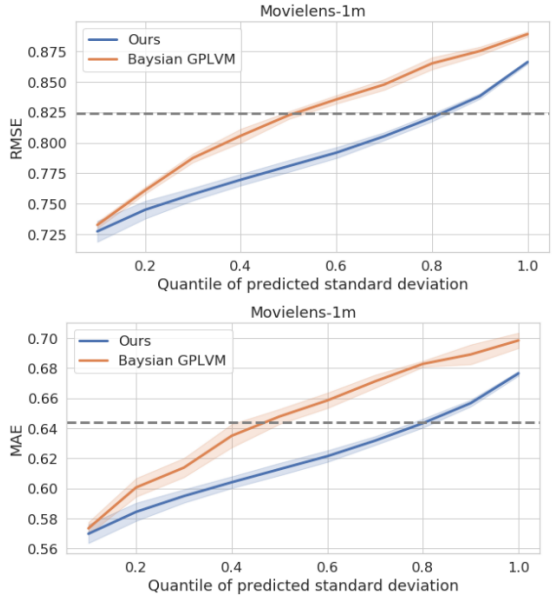
Scalable Gaussian Processes for collaborative filtering

- Our proposal: let f be a Gaussian Process regression and learn the embeddings jointly.

- Challenges and solutions:
  - Kernels → Define two kernels for $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively.
  - Scalability (up to 10 million) → SOTA scalable GP.
  - Inducing points → "*inducing pairs*", initialized via kernel PCA
  - Evaluation → Quantile-Performance-Plot



	RMSE	MAE
CF-NADE [Zheng et al., 2016]	0.829	-
GPLVM [Lawrence and Urtasun, 2009]	0.880	**0.644**
Sparse FC [Muller et al., 2018]	**0.824**	-
GC [Berg et al., 2017]	0.832	-
CWOCF [Lu et al., 2013]	0.958	0.761
LLORMA [Lee et al., 2013]	0.865	-
Biased MF	0.863	0.678
SVD++	0.893	0.705
Bayesian GPLVM at $q = 100\%$	0.889	0.698
MW-GP at $q = 100\%$	0.866	0.676
MW-GP at $q = 90\%$	0.838	0.657
MW-GP at $q = 80\%$	0.821	0.643



Yang and Buettner. Multi-output gaussian processes for uncertainty-aware recommender systems. *UAI*. 2021.

SIEMENS

Scalable Gaussian Processes for Deep Neural Networks

- CNN + Scalable GP:

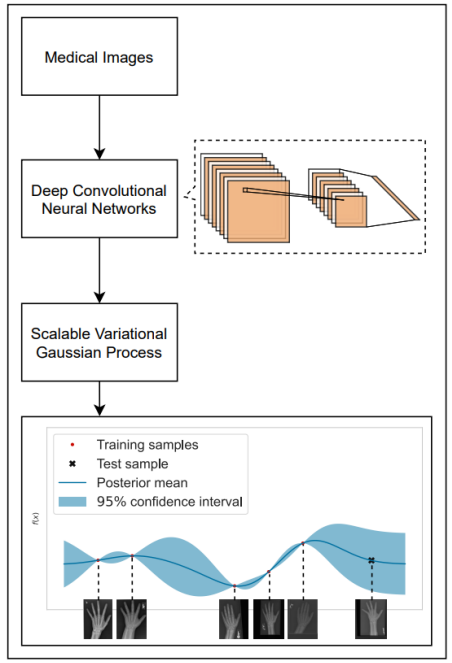


TABLE I
BONE AGE PREDICTION WITH DENSENET121

Output Layer	Transfer Learning	RMSE (No pre-training)	RMSE (DML)	RMSE (CAE)
Linear*	Yes	12.118 ± 0.277	11.667 ± 0.231	14.076 ± 0.281
SVGP†	Yes	11.697 ± 0.102	$\mathbf{11.440 \pm 0.132}$	$\mathbf{13.536 \pm 0.279}$
PPGP†	Yes	$\mathbf{11.679 \pm 0.061}$	11.529 ± 0.089	13.694 ± 0.274
Linear*	No	19.934 ± 0.246	$\mathbf{15.805 \pm 0.157}$	15.340 ± 0.390
SVGP†	No	$\mathbf{17.723 \pm 0.298}$	15.832 ± 0.284	$\mathbf{15.323 \pm 0.411}$
PPGP†	No	18.341 ± 0.234	16.084 ± 0.336	15.752 ± 0.352

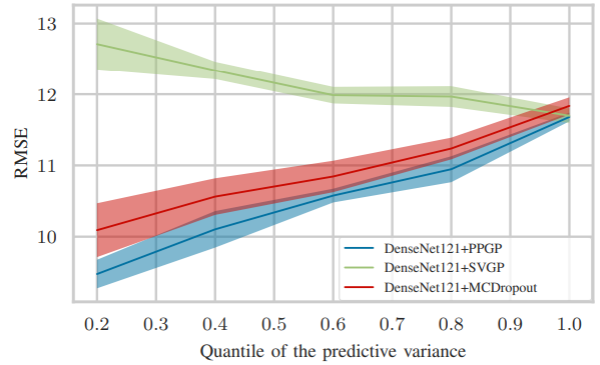
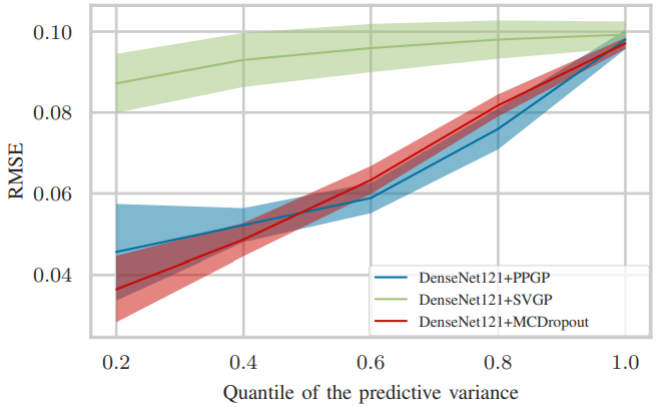


TABLE II
LESION LOCALIZATION WITH DENSENET121

Output Layer	Transfer Learning	RMSE (No pre-training)	RMSE (Metric)	RMSE (CAE)
Linear*	Yes	0.102 ± 0.002	0.101 ± 0.002	0.102 ± 0.003
SVGP†	Yes	0.099 ± 0.003	0.101 ± 0.003	0.104 ± 0.004
PPGP†	Yes	$\mathbf{0.098 \pm 0.002}$	$\mathbf{0.098 \pm 0.002}$	$\mathbf{0.099 \pm 0.002}$
Linear*	No	0.116 ± 0.001	0.114 ± 0.003	0.114 ± 0.002
SVGP†	No	0.118 ± 0.002	0.114 ± 0.003	0.112 ± 0.003
PPGP†	No	$\mathbf{0.115 \pm 0.002}$	$\mathbf{0.111 \pm 0.005}$	$\mathbf{0.110 \pm 0.002}$



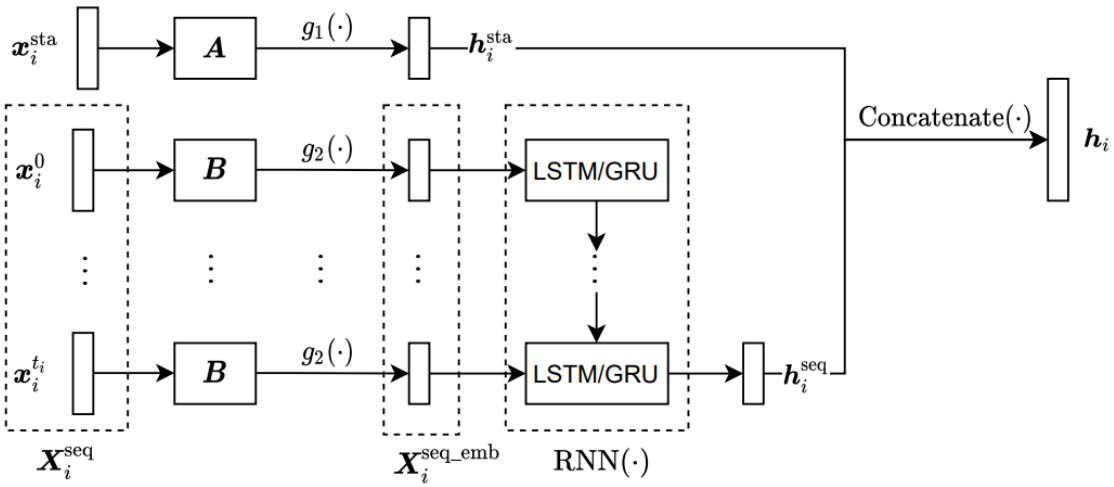
- Tasks: Bone-age prediction, lesion localization.
- Challenge: inducing points initialization.
- Solution: pre-training with auto-encoders.

Wu, et al. Quantifying predictive uncertainty in medical image analysis with deep kernel learning. *2021 IEEE ICHI*. 2021.

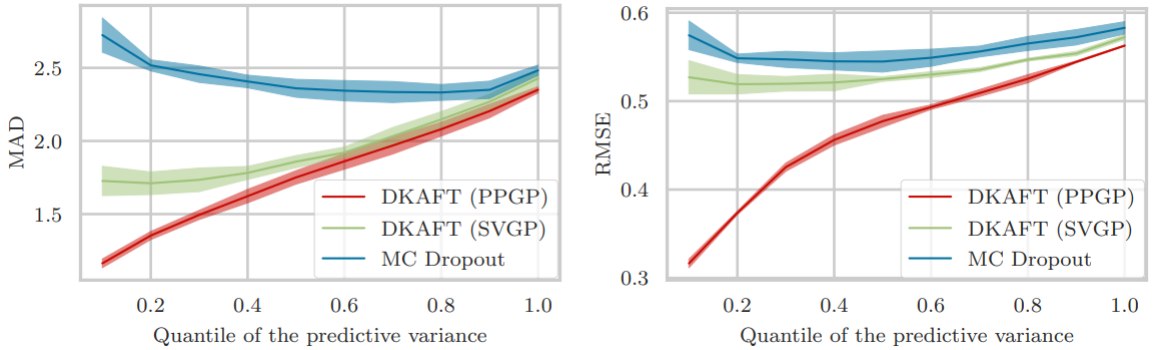
SIEMENS

Scalable Gaussian Processes for Deep Neural Networks

- RNN + Scalable GP



Method	Pre-training	Progression-Free Survival MAD	Progression-Free Survival RMSE	Length-of-Stay MAD	Length-of-Stay RMSE
Cox Regression	–*	200.800 ± 16.984	1.609 ± 0.054	2.727 ± 0.007	0.638 ± 0.0002
AFT Regression	–*	206.065 ± 8.988	1.685 ± 0.080	2.742 ± 0.014	0.630 ± 0.0001
RNN+AFT	None	150.918 ± 3.009	1.273 ± 0.019	2.476 ± 0.040	0.575 ± 0.003
DKAFT (ExactGP)	None	144.622 ± 8.689	1.225 ± 0.022	–†	–†
DKAFT (SVGP)	None	154.237 ± 13.490	1.211 ± 0.020	2.428 ± 0.056	0.572 ± 0.003
DKAFT (PPGP)	None	147.108 ± 6.284	1.220 ± 0.019	2.351 ± 0.021	0.563 ± 0.001
RNN+AFT	DML	138.155 ± 7.496	1.267 ± 0.007	2.452 ± 0.057	0.568 ± 0.001
DKAFT (ExactGP)	DML	$\mathbf{134.422 \pm 7.255}$	1.202 ± 0.012	–†	–†
DKAFT (SVGP)	DML	151.852 ± 11.305	1.221 ± 0.007	2.438 ± 0.079	0.567 ± 0.005
DKAFT (PPGP)	DML	146.616 ± 17.109	$\mathbf{1.195 \pm 0.008}$	$\mathbf{2.346 \pm 0.042}$	$\mathbf{0.557 \pm 0.002}$



- Tasks: Progression-free survival, length of stay [Sources]
- Challenge: inducing points initialization
- Solution: pre-training, esp. the deep metric learning:

$$\mathcal{J}_{\text{triplet}} = \sum_{i=1}^{n} \left[\mathrm{d}(\boldsymbol{h}_i^{\mathrm{A}}, \boldsymbol{h}_i^{\mathrm{P}}) - \mathrm{d}(\boldsymbol{h}_i^{\mathrm{A}}, \boldsymbol{h}_i^{\mathrm{N}}) + \alpha \right]_+,$$

Wu, et al. Uncertainty-Aware Time-to-Event Prediction using Deep Kernel Accelerated Failure Time Models. *MLHC (JMLR)*. 2021.

SIEMENS

Disclaimer

Subject to changes and errors. The information given in this document only contains general descriptions and/or performance features which may not always specifically reflect those described, or which may undergo modification in the course of further development of the products. The requested performance features are binding only when they are expressly agreed upon in the concluded contract.

Contact

Published by Siemens 2023

Dr. Yinchong Yang

Senior Key Expert Research Scientist
T DAI HCA-DE

Otto-Hahn-Ring 6
85579 München
Germany

SIEMENS