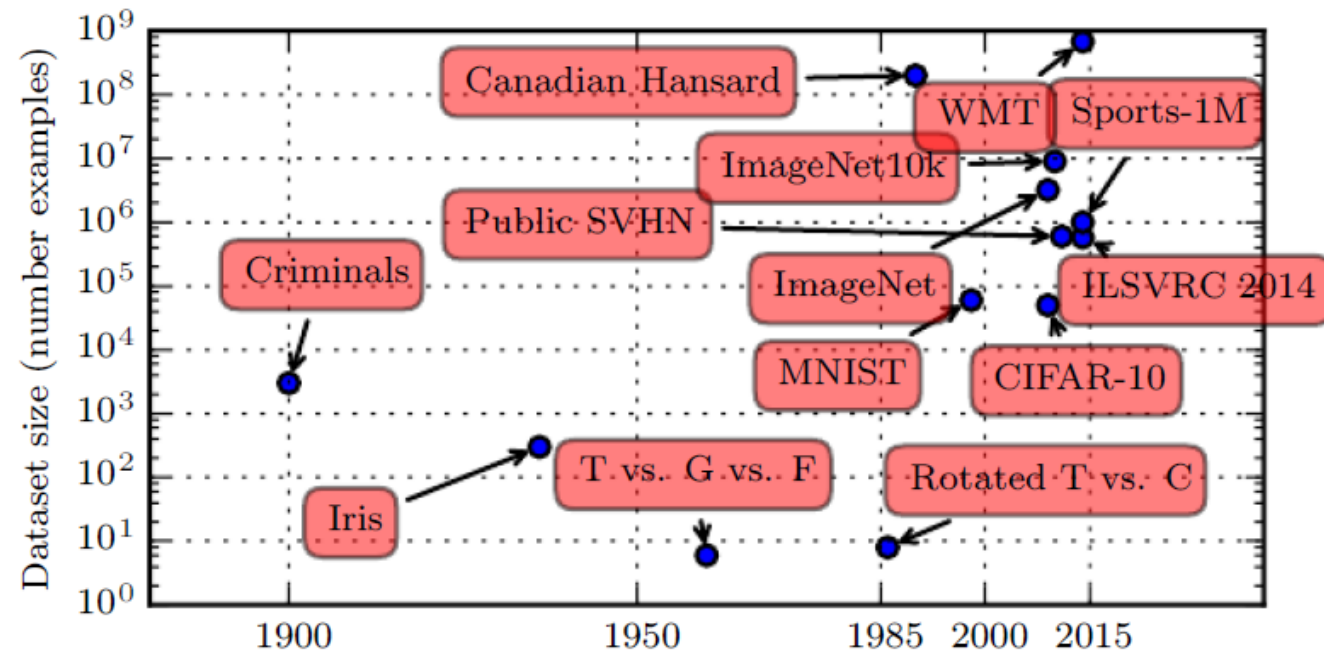


# Conditional Neural Processes

RG Presentation

# Motivation – Data Efficiency

- Great success has been seen with training supervised models with large datasets
- However, these successes have not translated well to the low data regime



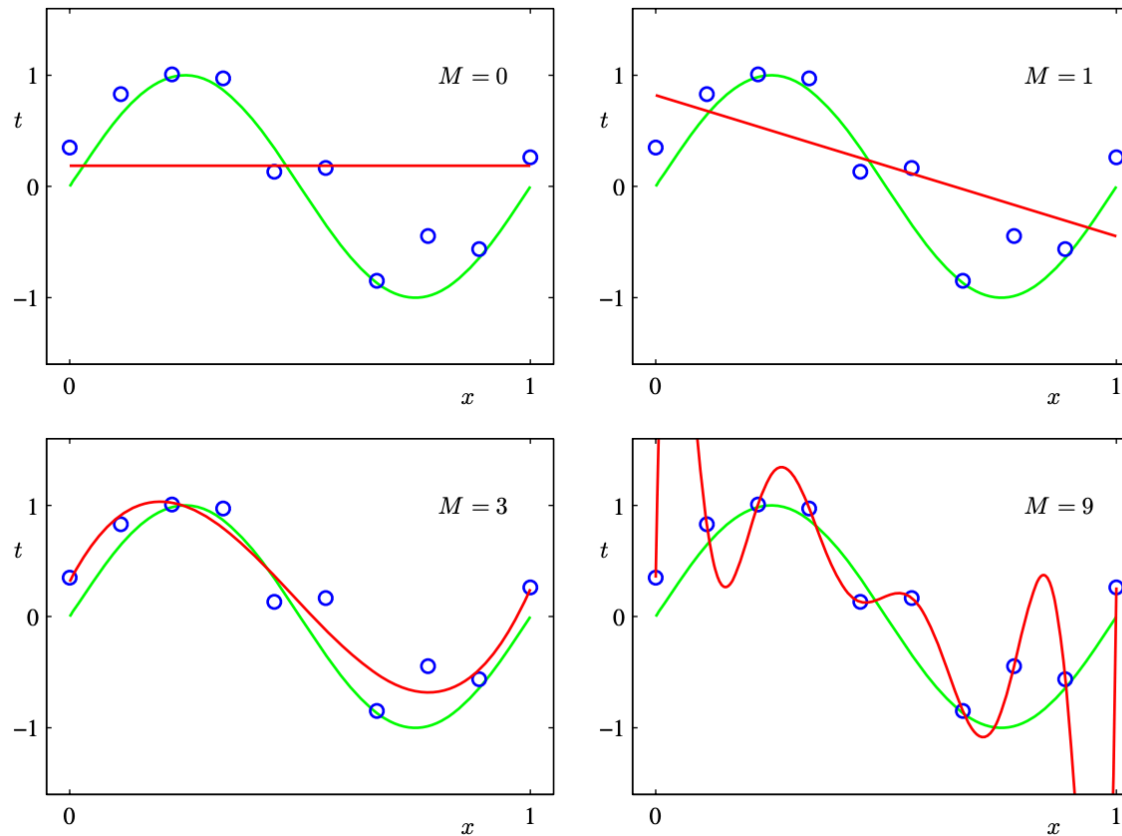
# Motivation – Data Efficiency

Some possible approaches to this problem

- Meta-learning (learning to extract common structure between multiple tasks/datasets so only a few examples are needed to adapt to new tasks)
- Active Learning (using humans/oracles in the training loop to only label what is necessary)
- Pre-training (training a model on an auxiliary task with unlabeled data)
- Gaussian Processes (Explicitly defining a distribution over functions)

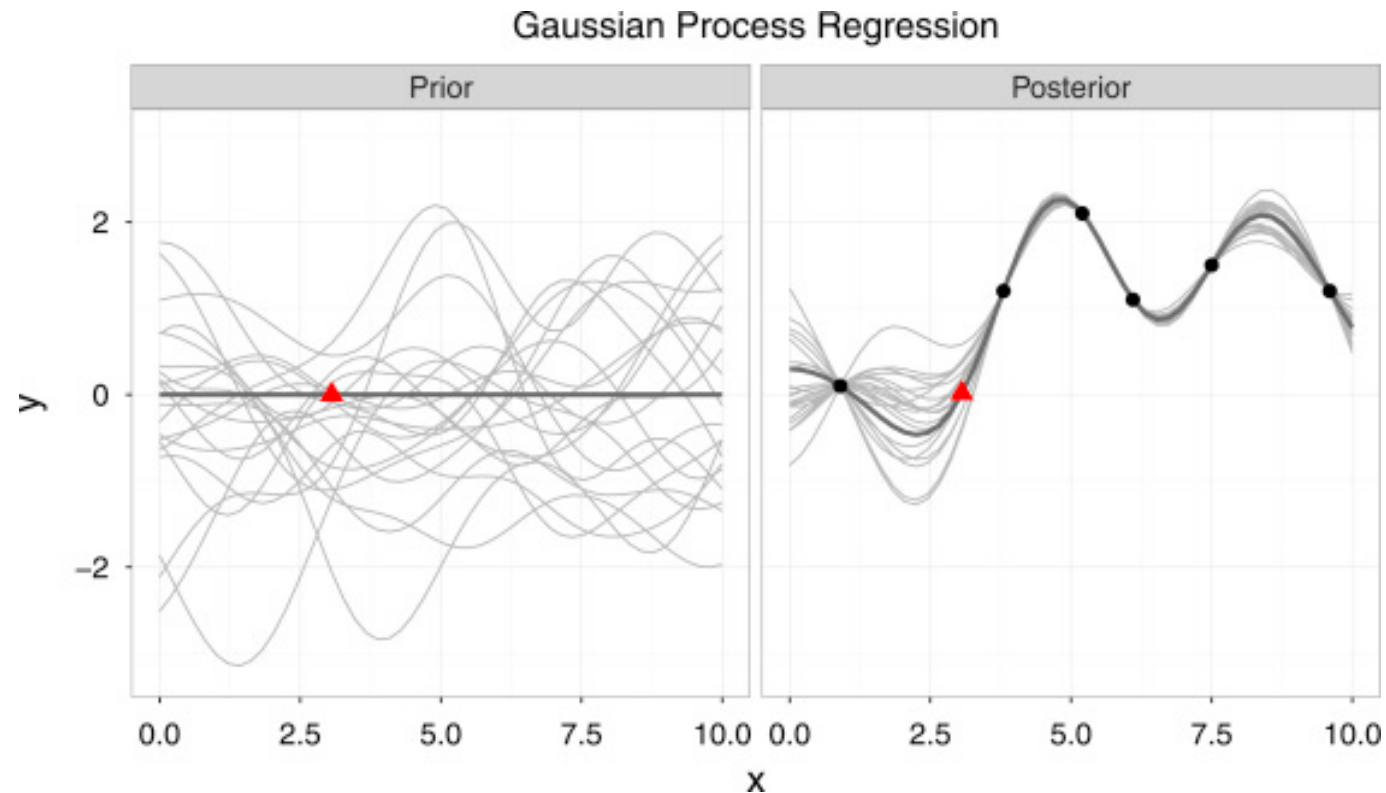
# Motivation – Uncertainty Estimates

Working in a low data regime necessitates uncertainty estimates



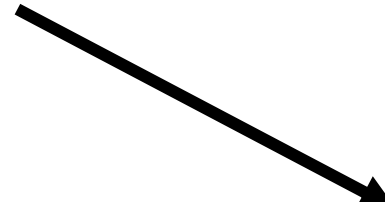
# Motivation – Uncertainty Estimates

GPs (Naturally provide uncertainty estimates in their posteriors)



# What's wrong with GPs then?

- $O(N^3)$  time complexity and  $O(N^2)$  space complexity

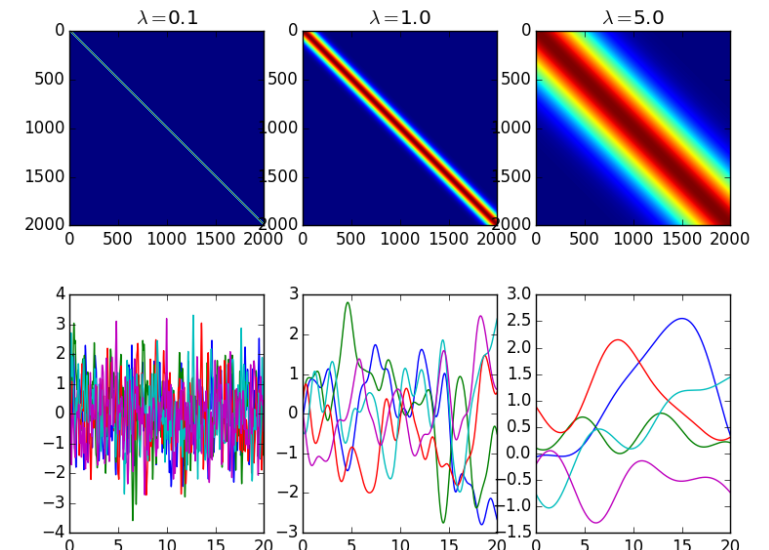


$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}[\mathbf{f}_* | X, \mathbf{y}, X_*] = K(X_*, X) \underline{[K(X, X) + \sigma_n^2 I]^{-1}} \mathbf{y},$$

$$\text{cov}(\mathbf{f}_*) = K(X_*, X_*) - K(X_*, X) \underline{[K(X, X) + \sigma_n^2 I]^{-1}} K(X, X_*).$$

# Similarities between meta-learning and GPs

- Both utilize common structure between train and test data to achieve data efficiency
  - Meta-learning (structure extracted via training on multiple tasks)
  - GPs (structure of function defined by covariance function, usually selected by testing different functions and selecting via marginal likelihood)



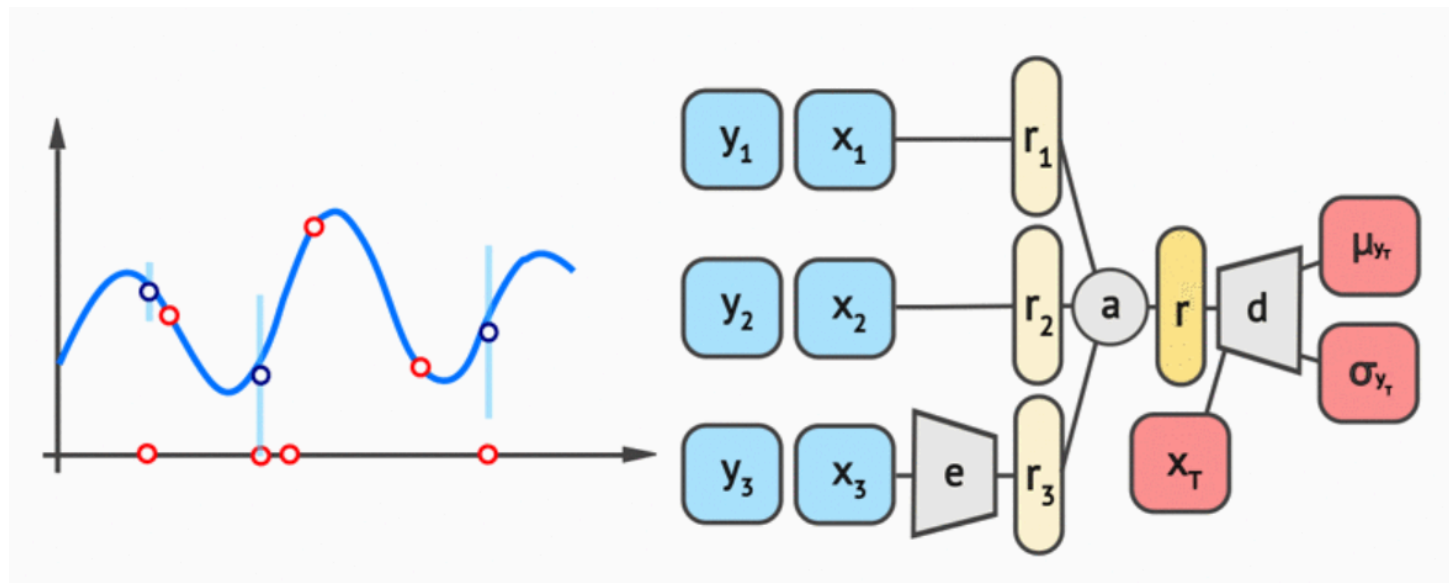
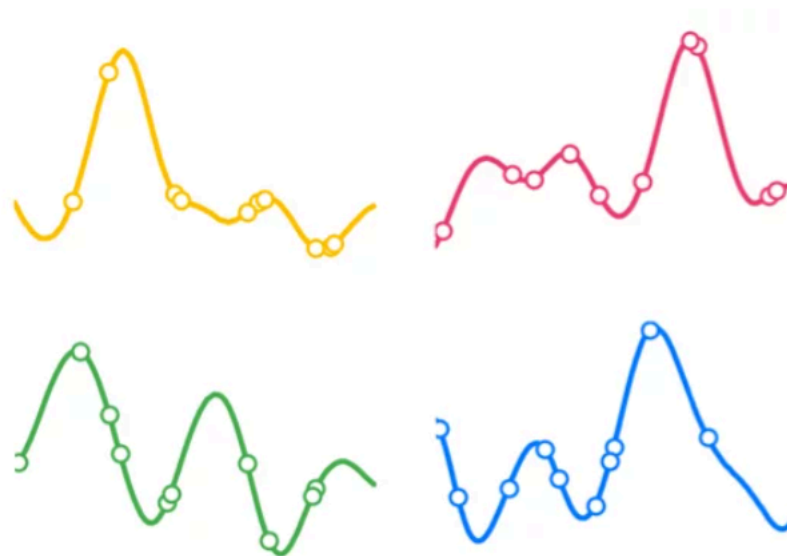
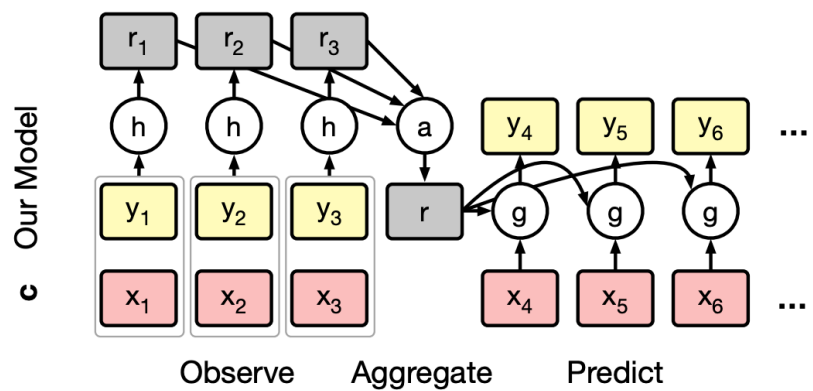
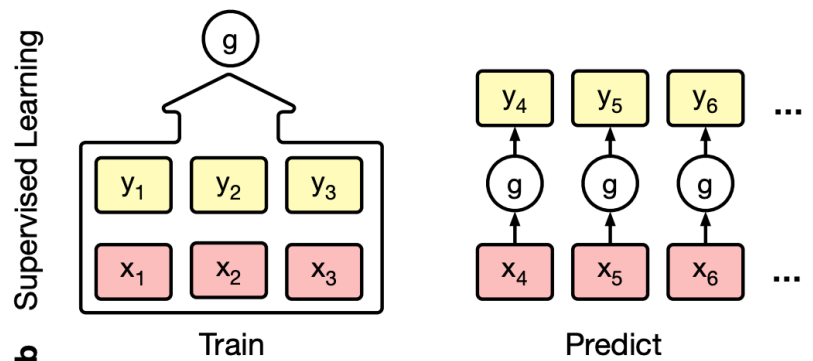
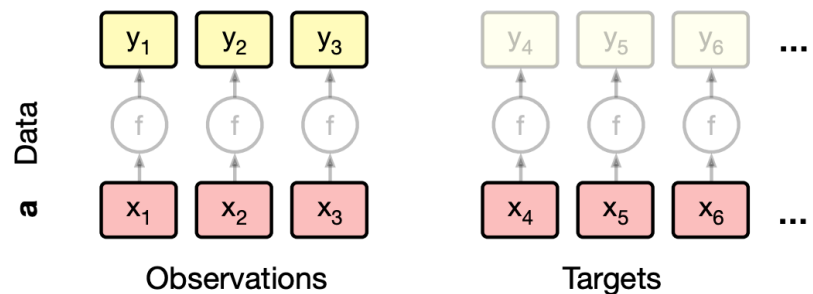
# The Idea

Rather than trying to get exact analytic posteriors with GPs, use neural networks to learn distributions over functions (inference is very cheap)

Neural processes = meta-learning + model outputting distribution parameters (rather than point estimate)



# Neural processes

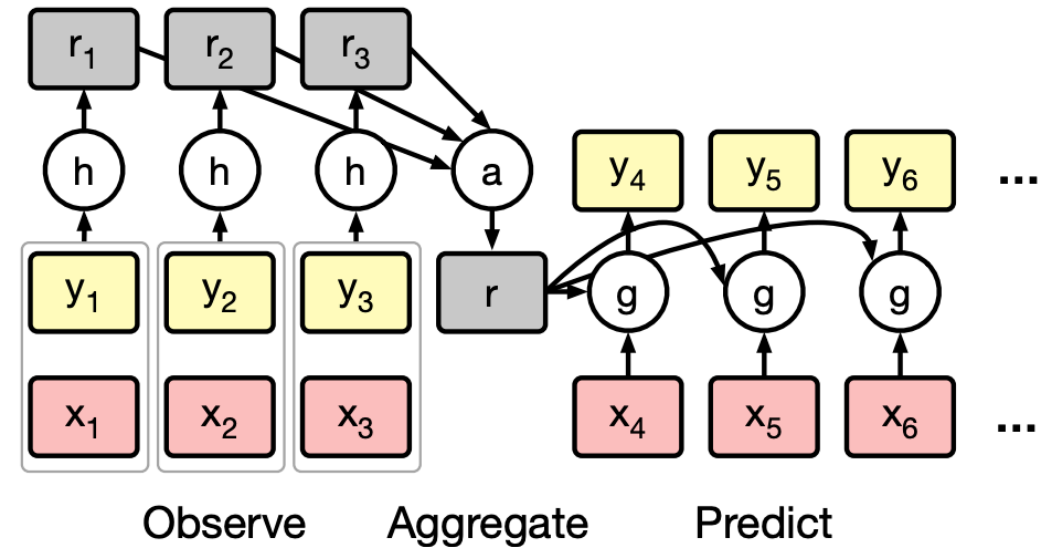


# Permutation Invariance

Encoder should output the same embedding ( $r$ ), regardless of ordering of observed data

Aggregate operator should be permutation invariant

This paper just uses the mean



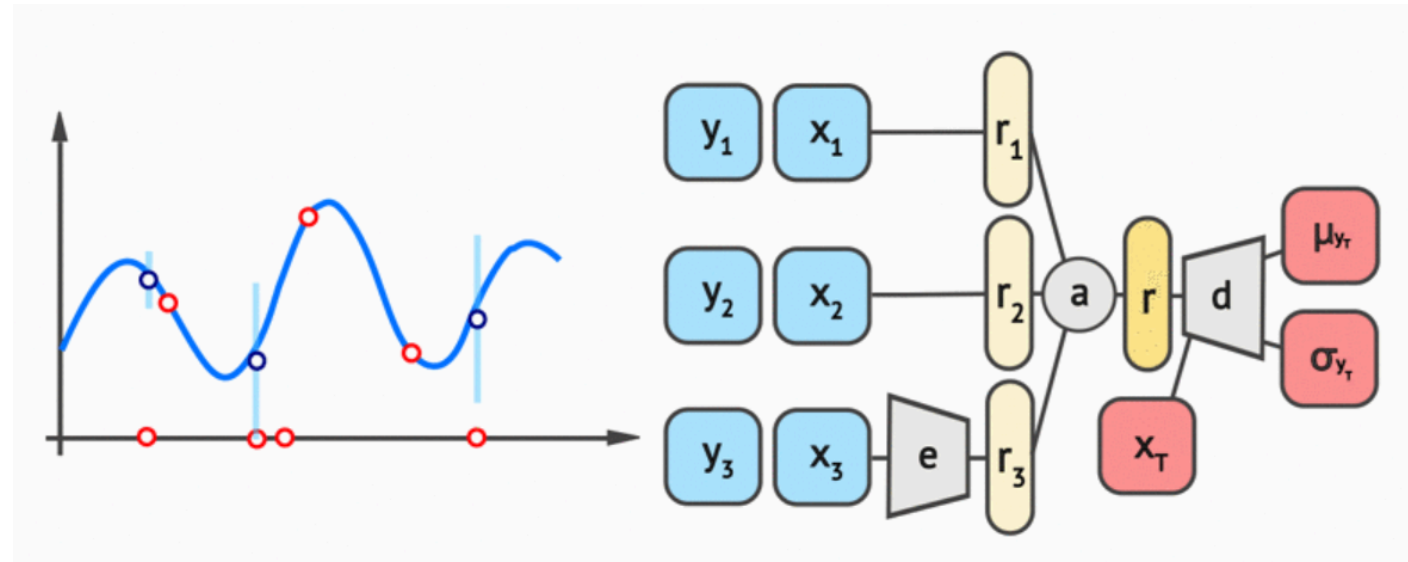
Follow-up papers use attention weighted sum over embeddings

# Loss

- Since we are outputting the parameters of a distribution (gaussian in figure), we can just use the maximum likelihood objective to train the model

$$\mathcal{L}(\theta) = -\mathbb{E}_{f \sim P} \left[ \mathbb{E}_N \left[ \log Q_{\theta}(\{y_i\}_{i=0}^{n-1} | O_N, \{x_i\}_{i=0}^{n-1}) \right] \right]$$

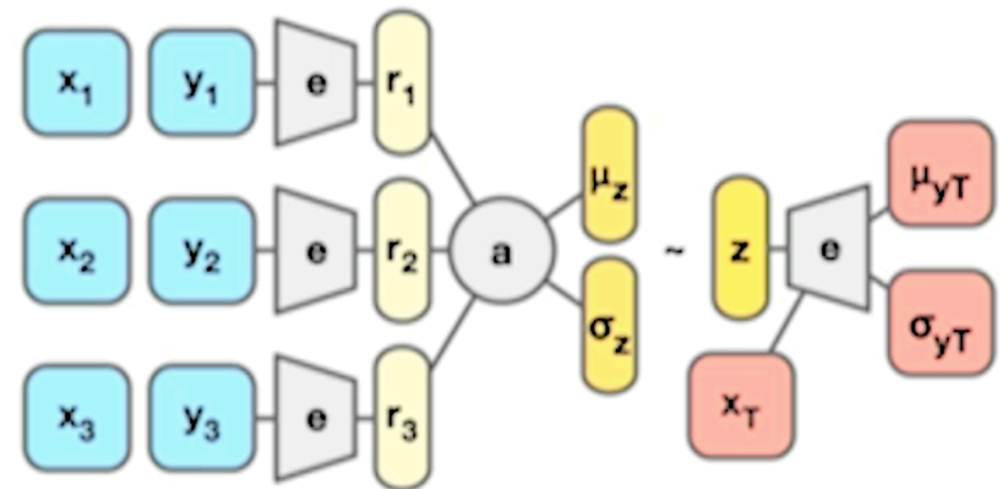
Thus, the targets it scores  $Q_{\theta}$  on include *both* the observed and unobserved values. In practice, we take Monte Carlo estimates of the gradient of this loss by sampling  $f$  and  $N$ .



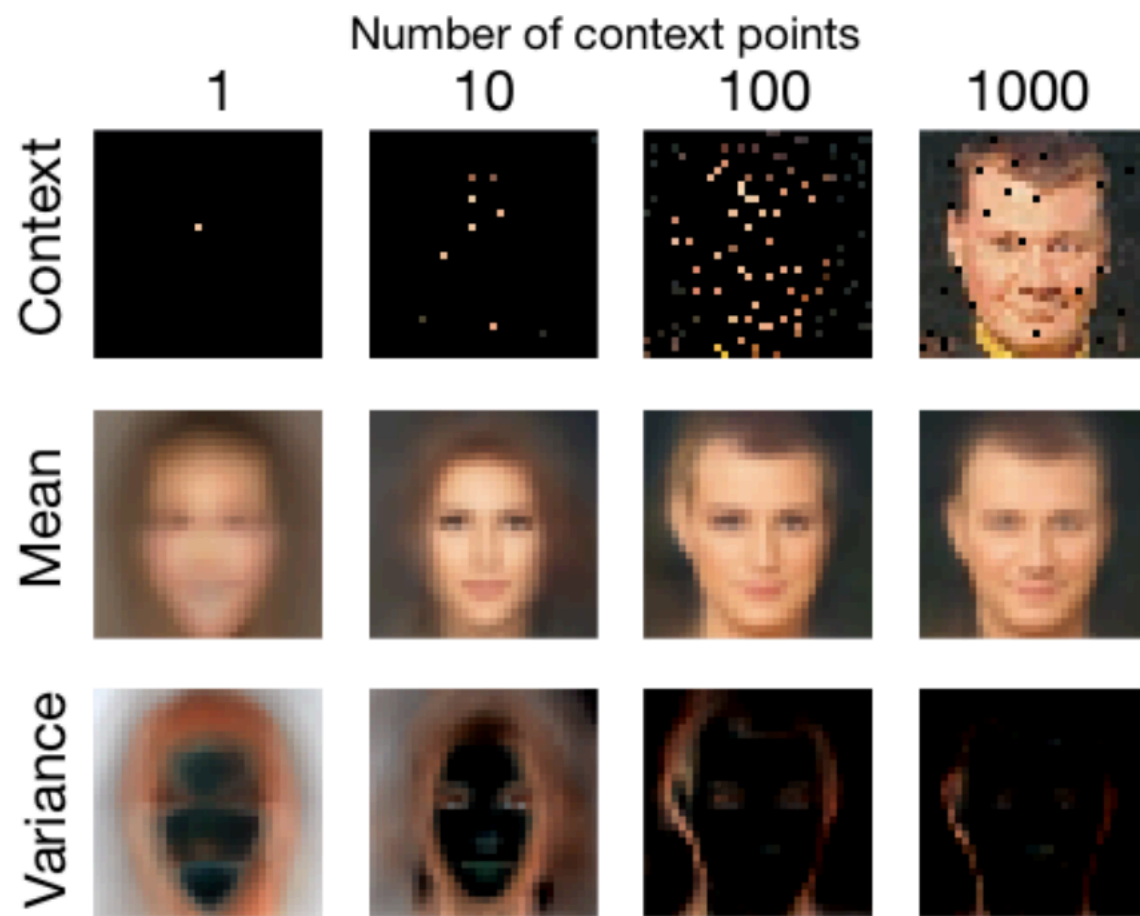
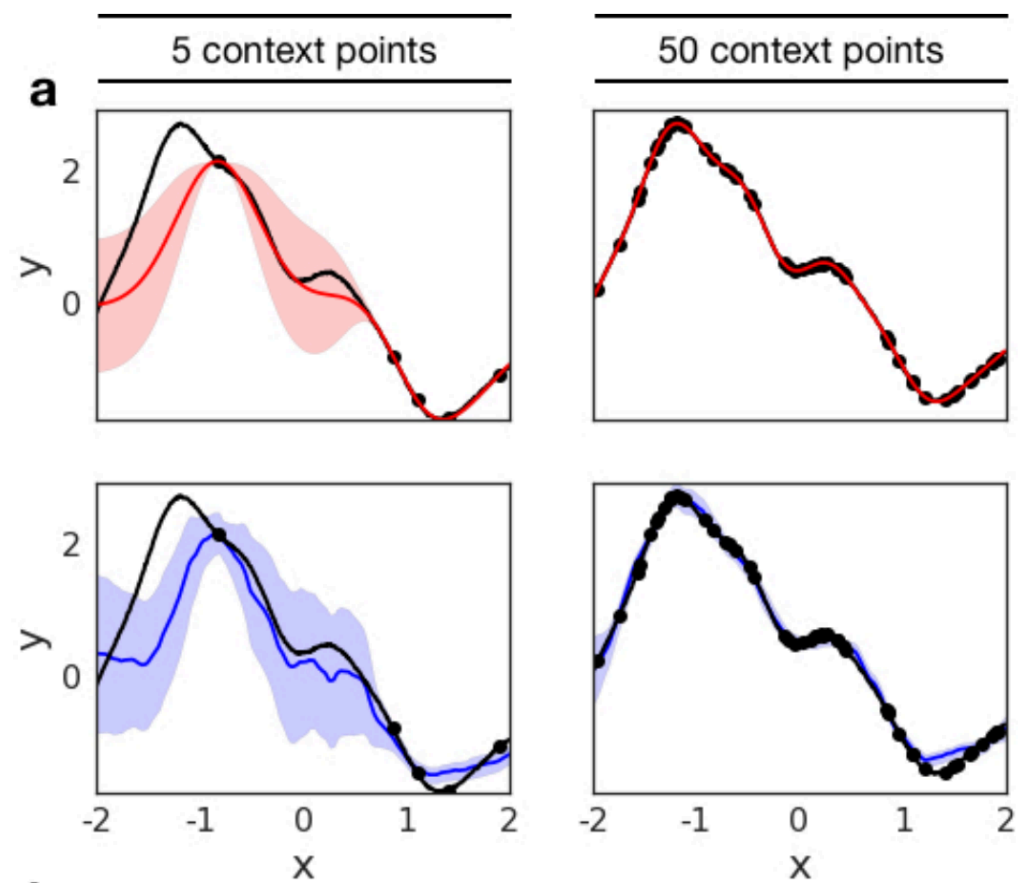
# Issues

- Does not produce coherent samples (model learns factored prediction of means and variances, disregarding covariances between points, thus samples are very noisy)
- This problem was fixed by introducing a latent variable (captures global information)

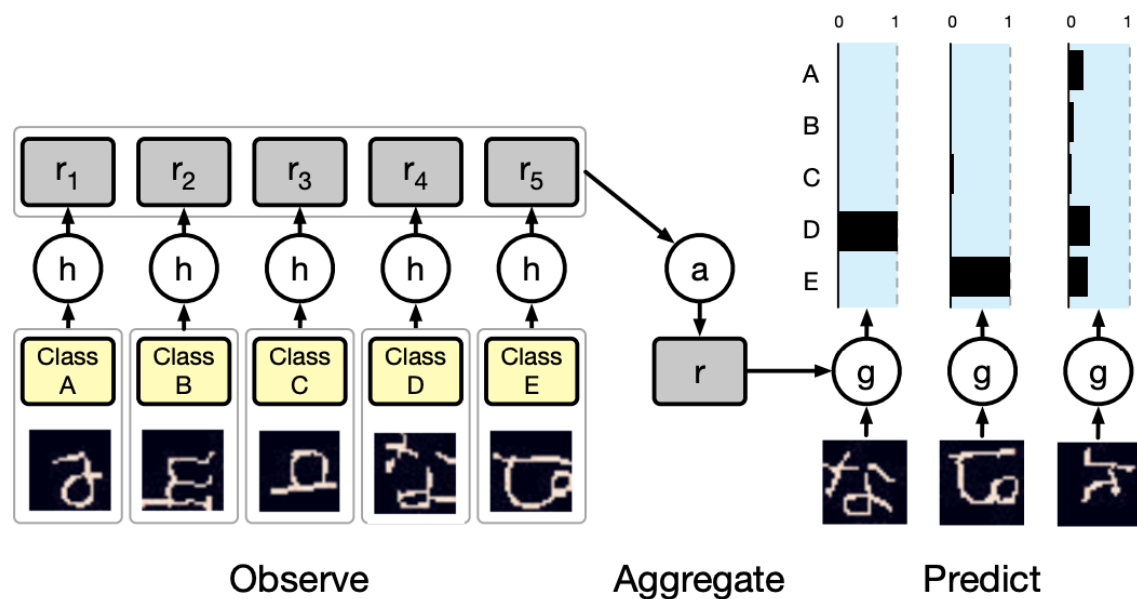
This is the “Neural Processes” paper  
Optimise ELBO instead (or another objective they introduce in that paper)



# Experiments



# Experiments



	5-way Acc		20-way Acc		Runtime
	1-shot	5-shot	1-shot	5-shot	
MANN	82.8%	94.9%	-	-	$\mathcal{O}(nm)$
MN	<b>98.1%</b>	<b>98.9%</b>	<b>93.8%</b>	<b>98.5%</b>	$\mathcal{O}(nm)$
CNP	95.3%	98.5%	89.9%	96.8%	$\mathcal{O}(n + m)$

# Questions

- Have people explored meta-learning + deep kernel learning for GPs?
- What other invariances might we want our model to have?