Bayesian Deep Learning and Uncertainty

Christos Louizos, Max Welling

Amsterdam Machine Learning Lab

University of Amsterdam

Overview

- Part 1:
  - Structured approximate posteriors in Bayesian neural nets
- Part 2:
  - Exploring model uncertainty

Part 1

Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors

ICML 2016

VI for Bayesian neural networks (BNNs)

VI for Bayesian neural networks (BNNs)

Stochastic gradient VI for BNNs optimizes:

$$\mathbb{E}_{\prod_{i=1}^{H} q(\mathbf{W}_i)}[\log p(\mathbf{y}|\mathbf{x}, \mathbf{W}_{1:H})] - \sum_{i=1}^{H} KL(q(\mathbf{W}_i), p(\mathbf{W}_i))$$

w.r.t. to the posterior distributions q

VI for Bayesian neural networks (BNNs)

Stochastic gradient VI for BNNs optimizes:

$$\mathbb{E}_{\prod_{i=1}^{H} q(\mathbf{W}_i)}[\log p(\mathbf{y}|\mathbf{x}, \mathbf{W}_{1:H})] - \sum_{i=1}^{H} KL(q(\mathbf{W}_i), p(\mathbf{W}_i))$$

w.r.t. to the posterior distributions q

Usual choices for q are fully factorized Gaussians

$$q(\mathbf{W}_i) = \prod_{r=1}^{R} \prod_{c=1}^{C} \mathcal{N}(\mu_{rc}, \sigma_{rc}^2)$$

Implications of fully factorized posteriors

Implications of fully factorized posteriors

- Doubles the amount of trainable parameters
  - Prohibitive for large neural networks

Implications of fully factorized posteriors

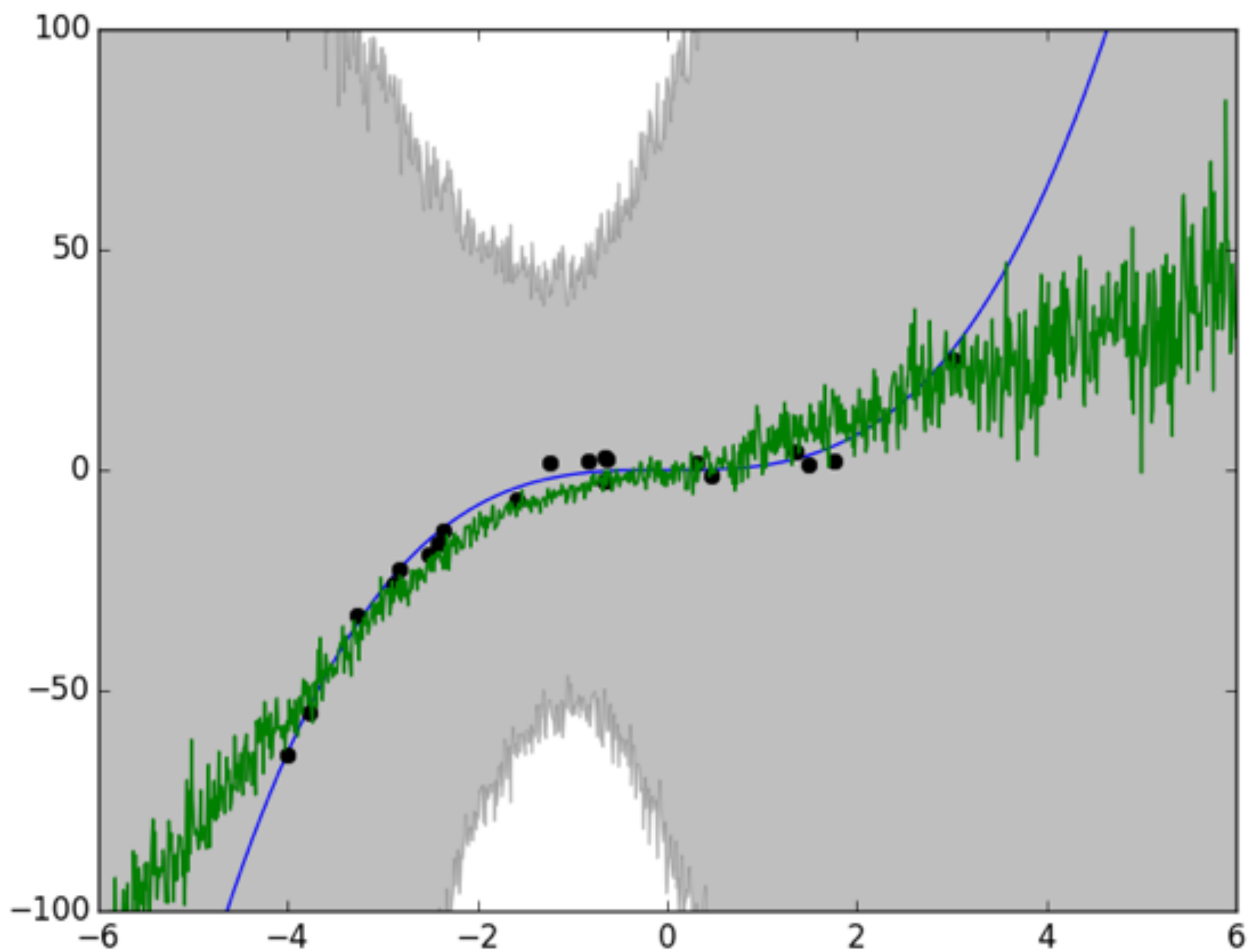
- Doubles the amount of trainable parameters
  - Prohibitive for large neural networks
- Too much “noise” due to a lot of independent sources of variation

Implications of fully factorized posteriors

- Doubles the amount of trainable parameters
  - Prohibitive for large neural networks
- Too much “noise” due to a lot of independent sources of variation
- No information sharing across weights

Implications of fully factorized posteriors

- Doubles
  - Prohibi
- Too muc urces of variatior
- No infor



Fully factorized VI BNN

Structured VI for BNNs

How we can introduce structure efficiently?

Structured VI for BNNs

How we can introduce structure efficiently?

Simple idea

Instead of approximating the marginal of each element in the weight matrix approximate the weight matrix directly!

Structured VI for BNNs

How we can introduce structure efficiently?

<u>Simple idea</u>

Instead of approximating the marginal of each element in the weight matrix approximate the weight matrix directly!

Assume a matrix Gaussian posterior over each weight matrix:

$$\begin{aligned} q(\mathbf{W}_i) &= \mathcal{MN}(\mathbf{M}_{r\times c}, \mathbf{U}_{r\times r}, \mathbf{V}_{c\times c}) \\ &= \mathcal{N}(vec(\mathbf{M}), \mathbf{U}\otimes\mathbf{V}) \end{aligned}$$

M corresponds to the mean matrix

U corresponds to the row covariance

V corresponds to the column covariance

Structured VI with matrix Gaussian posteriors

Structured VI with matrix Gaussian posteriors

- Compactly model correlations among weights

Structured VI with matrix Gaussian posteriors

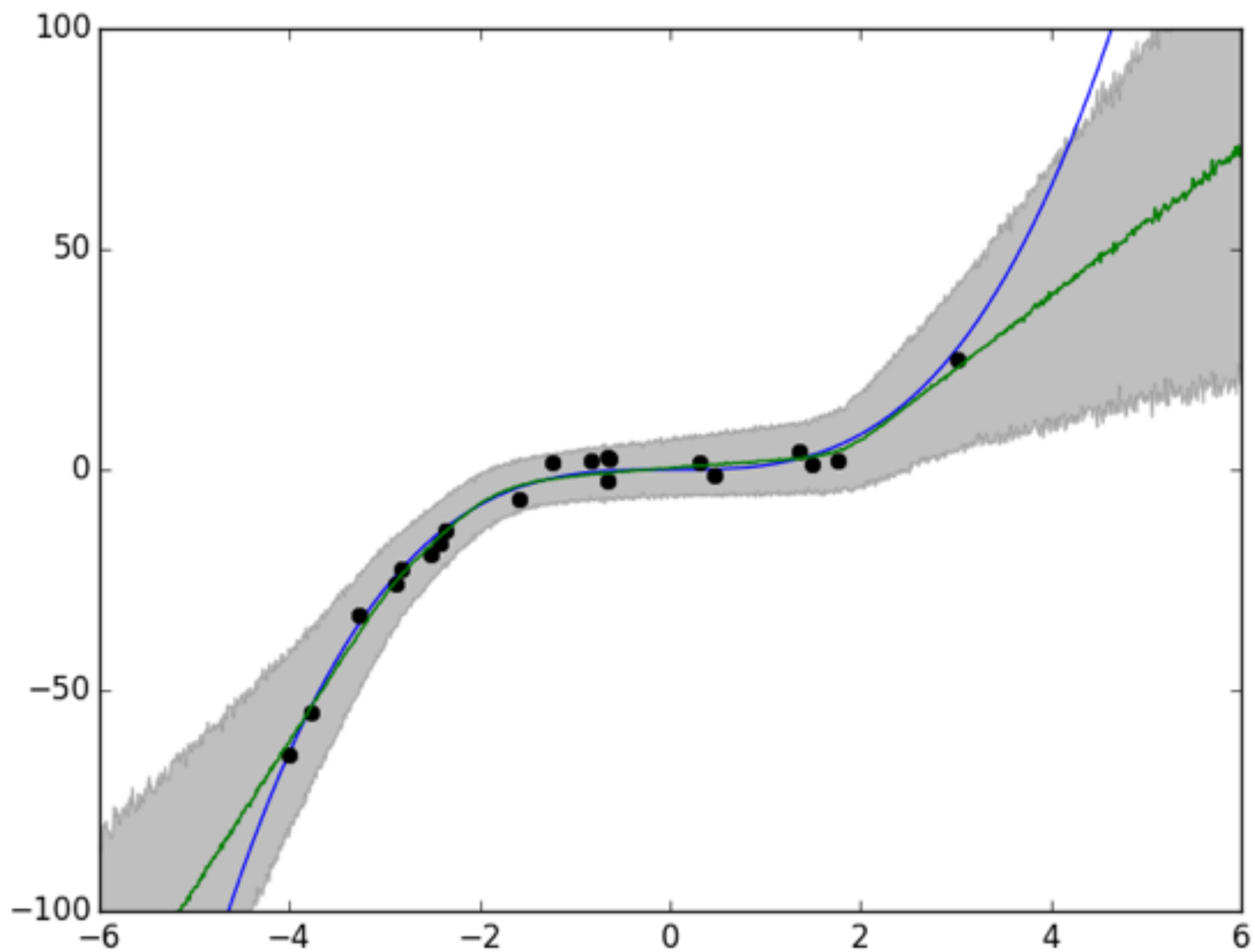
- Compactly model correlations among weights
- “Full” matrix Gaussian is still expensive

Structured VI with matrix Gaussian posteriors

- Compactly model correlations among weights
- "Full" matrix Gaussian is still expensive
- Low rank approximations for **U**, **V**
  - Diagonal approximations reduce parameters greatly
  - Still maintain some correlations among weights

Structured VI with matrix Gaussian posteriors

- Compa
- "Full" m
- Low rank
    - Diagor
    - Still ma

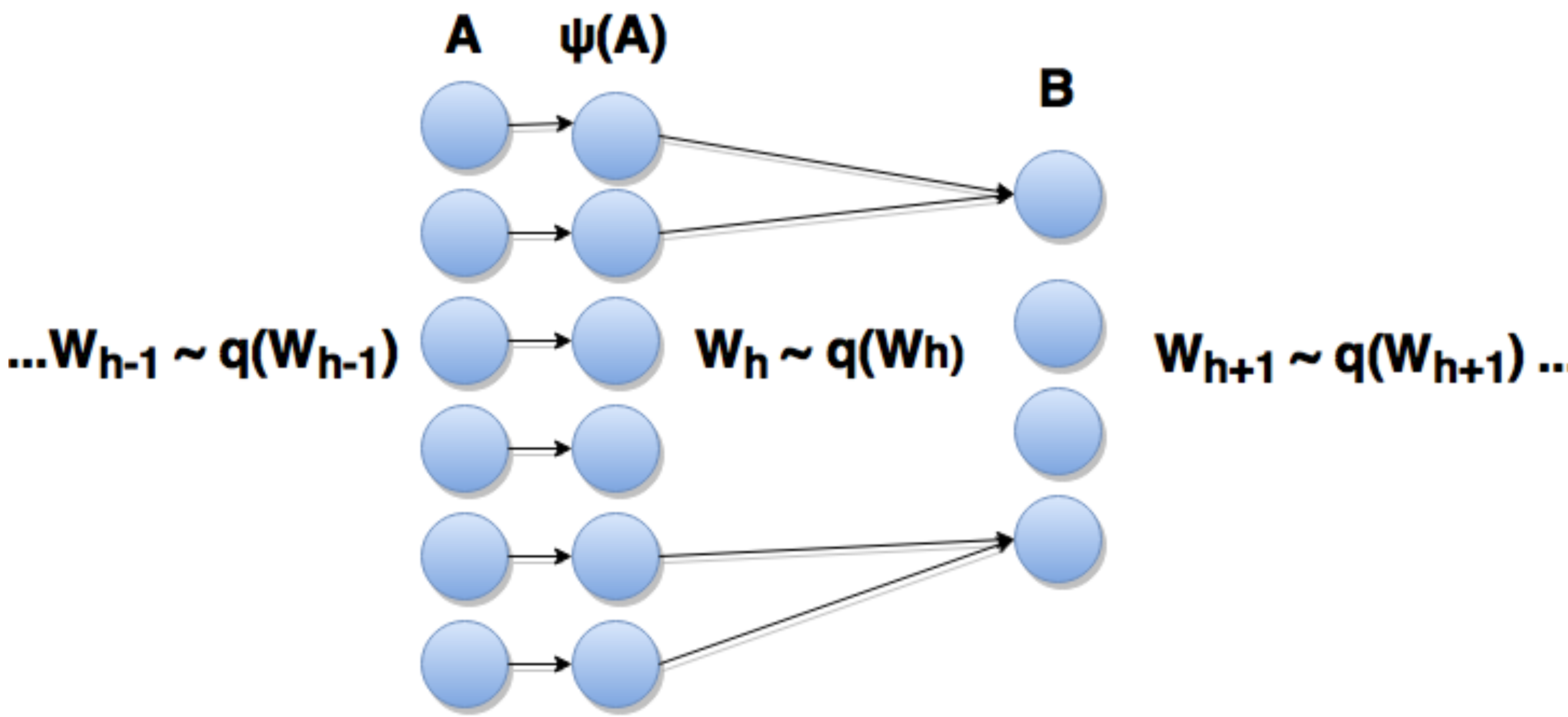


Structured VI BNN

Global posterior sampling

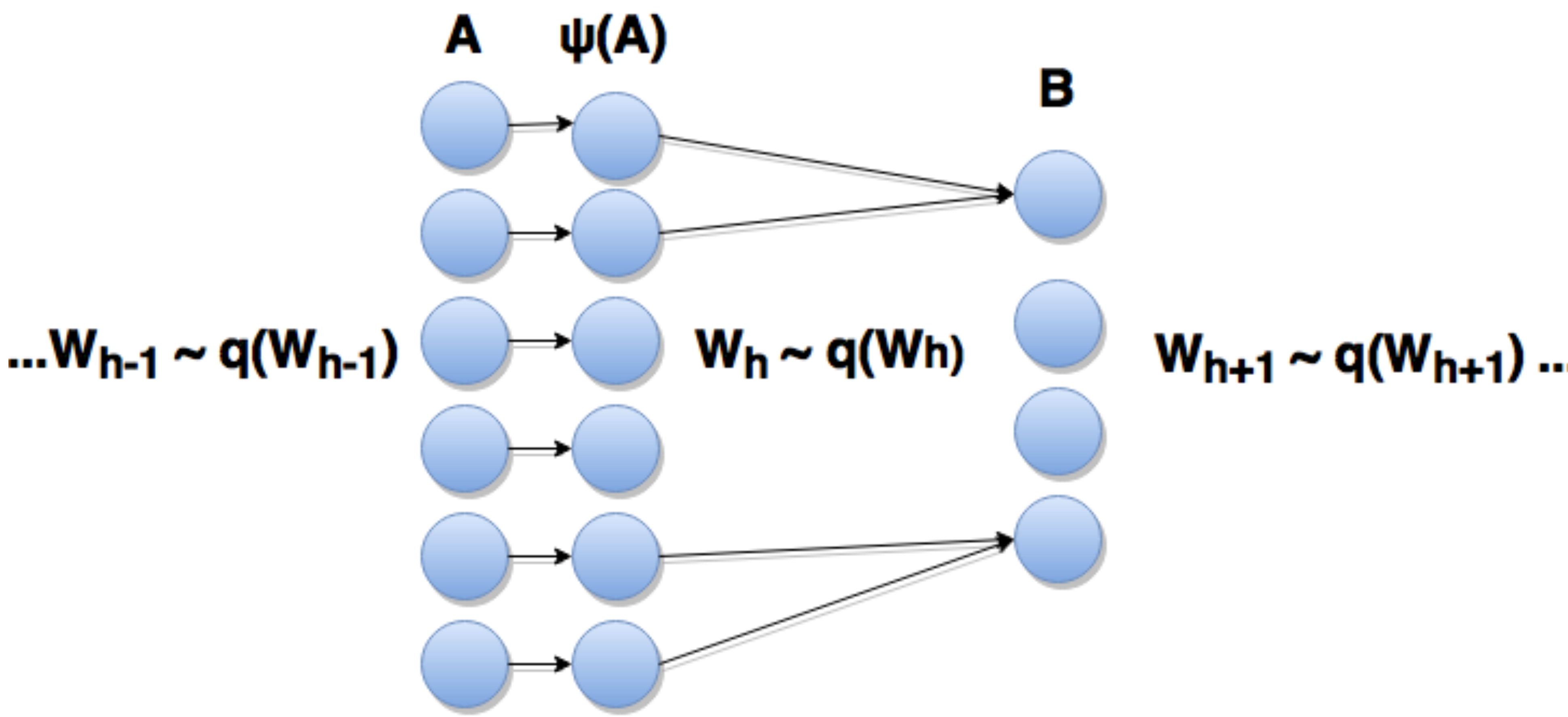
Global posterior sampling

In order to perform stochastic gradient VI we have to sample the approximate weight posterior at each layer:



Global posterior sampling

In order to perform stochastic gradient VI we have to sample the approximate weight posterior at each layer:



where **A** is a mini batch of the previous layer output, **ψ**(.) is an element wise nonlinearity and **B** is the layer output

Local posterior sampling

Local posterior sampling

[1] showed that we can obtain a lower variance gradient estimator if we instead sample the pre-activation variables **B** by locally marginalising the weights, i.e:

[1] “Variational Dropout and the local reparametrization trick”, Diederik P. Kingma, Tim Salimans, Max Welling, NIPS 2015

Local posterior sampling

[1] showed that we can obtain a lower variance gradient estimator if we instead sample the pre-activation variables **B** by locally marginalising the weights, i.e:

$$\mathbf{W}_h \sim \prod_{r=1}^{R} \prod_{c=1}^{C} \mathcal{N}(\mu_{rc}, \sigma_{rc}^2)$$

$$\mathbf{B} = \psi(\mathbf{A})\mathbf{W}_h$$

[1] “Variational Dropout and the local reparametrization trick”, Diederik P. Kingma, Tim Salimans, Max Welling, NIPS 2015

Local posterior sampling

[1] showed that we can obtain a lower variance gradient estimator if we instead sample the pre-activation variables **B** by locally marginalising the weights, i.e:

$$\mathbf{W}_h \sim \prod_{r=1}^{R} \prod_{c=1}^{C} \mathcal{N}(\mu_{rc}, \sigma_{rc}^2)$$

$$\mathbf{B} = \psi(\mathbf{A})\mathbf{W}_h$$

is equivalent to:

[1] “Variational Dropout and the local reparametrization trick”, Diederik P. Kingma, Tim Salimans, Max Welling, NIPS 2015

Local posterior sampling

[1] showed that we can obtain a lower variance gradient estimator if we instead sample the pre-activation variables **B** by locally marginalising the weights, i.e:

$$\mathbf{W}_h \sim \prod_{r=1}^{R} \prod_{c=1}^{C} \mathcal{N}(\mu_{rc}, \sigma_{rc}^2)$$

$$\mathbf{B} = \psi(\mathbf{A})\mathbf{W}_h$$

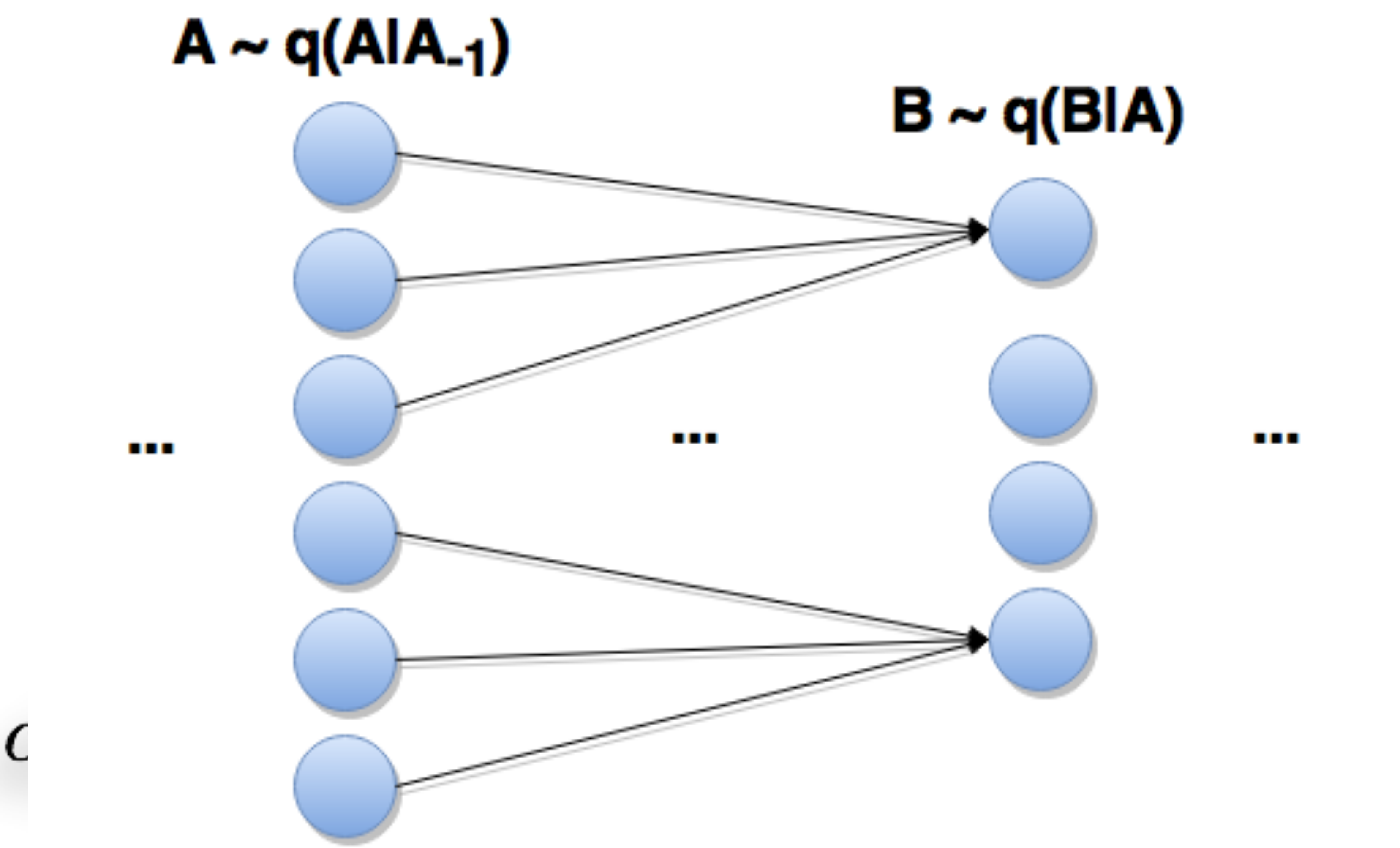
is equivalent to:

$$\mathbf{B}[r,c] \sim \mathcal{N}\left(\sum_c \psi(\mathbf{A})[r,c]\mu_{rc}, \sum_c \psi(\mathbf{A})[r,c]^2 \sigma_{rc}^2 \right)$$

[1] "Variational Dropout and the local reparametrization trick", Diederik P. Kingma, Tim Salimans, Max Welling, NIPS 2015

Local posterior sampling

[1] showed that we can obtain a lower variance gradient estimator if we instead sample the pre-activation variables **B** by locally marginalising the weights, i.e:



$\mathbf{B}[r,$ … $^{2}\sigma_{rc}^{2}\Big)$

[1] “Variational Dropout and the local reparametrization trick”, Diederik P. Kingma, Tim Salimans, Max Welling, NIPS 2015

Local posterior sampling with a Matrix Gaussian

Local posterior sampling with a Matrix Gaussian

Similarly we have that:

$$\mathbf{W}_h \sim \mathcal{MN}(\mathbf{M}_h, \mathbf{U}_h, \mathbf{V}_h)$$

$$\mathbf{B} = \psi(\mathbf{A})\mathbf{W}_h$$

Local posterior sampling with a Matrix Gaussian

Similarly we have that:

$$\mathbf{W}_h \sim \mathcal{MN}(\mathbf{M}_h, \mathbf{U}_h, \mathbf{V}_h)$$

$$\mathbf{B} = \psi(\mathbf{A})\mathbf{W}_h$$

is equivalent to:

$$\mathbf{B} \sim \mathcal{MN}(\psi(\mathbf{A})\mathbf{M}_h, \psi(\mathbf{A})\mathbf{U}_h\psi(\mathbf{A})^T, \mathbf{V}_h)$$

Local posterior sampling with a Matrix Gaussian

Similarly we have that:

$$\mathbf{W}_h \sim \mathcal{MN}(\mathbf{M}_h, \mathbf{U}_h, \mathbf{V}_h)$$

$$\mathbf{B} = \psi(\mathbf{A})\mathbf{W}_h$$

is equivalent to:

$$\mathbf{B} \sim \mathcal{MN}(\psi(\mathbf{A})\mathbf{M}_h, \psi(\mathbf{A})\mathbf{U}_h\psi(\mathbf{A})^T, \mathbf{V}_h)$$

or else:

$$\mathbf{B} \sim \mathcal{N}(vec(\psi(\mathbf{A})\mathbf{M}_h), \mathbf{K}(\mathbf{A}, \mathbf{A}))$$

$$\mathbf{K}(\mathbf{A}, \mathbf{A}) = \mathbf{K}_{out} \otimes \mathbf{K}_{in}(\mathbf{A}, \mathbf{A})$$

Local posterior sampling with a Matrix Gaussian

Similarly we have that:

$$\mathbf{W}_h \sim \mathcal{MN}(\mathbf{M}_h, \mathbf{U}_h, \mathbf{V}_h)$$

$$\mathbf{B} = \psi(\mathbf{A})\mathbf{W}_h$$

is equivalent to:

$$\mathbf{B} \sim \mathcal{MN}(\psi(\mathbf{A})\mathbf{M}_h, \psi(\mathbf{A})\mathbf{U}_h\psi(\mathbf{A})^T, \mathbf{V}_h)$$

or else:

$$\mathbf{B} \sim \mathcal{N}(vec(\psi(\mathbf{A})\mathbf{M}_h), \mathbf{K}(\mathbf{A}, \mathbf{A}))$$

$$\mathbf{K}(\mathbf{A}, \mathbf{A}) = \mathbf{K}_{out} \otimes \mathbf{K}_{in}(\mathbf{A}, \mathbf{A})$$

where:

$$\mathbf{K}_{in}(\mathbf{A}, \mathbf{A}) = \psi(\mathbf{A})\mathbf{U}_h\psi(\mathbf{A})^T$$

$$\mathbf{K}_{out} = \mathbf{V}_h$$

Local posterior sampling with a Matrix Gaussian

Similarly we have that:

So we can view each layer as a finite rank multi-output Gaussian Process and the whole network as a deep Gaussian Process[2] with a specific kernel!

or else:

$$\mathbf{B} \sim \mathcal{N}(vec(\psi(\mathbf{A})\mathbf{M}_h), \mathbf{K}(\mathbf{A}, \mathbf{A}))$$

$$\mathbf{K}(\mathbf{A}, \mathbf{A}) = \mathbf{K}_{out} \otimes \mathbf{K}_{in}(\mathbf{A}, \mathbf{A})$$

where:

$$\mathbf{K}_{in}(\mathbf{A}, \mathbf{A}) = \psi(\mathbf{A})\mathbf{U}_h\psi(\mathbf{A})^T$$

$$\mathbf{K}_{out} = \mathbf{V}_h$$

[2] “Deep Gaussian Processes”, Andreas C. Damianou, Neil D. Lawrence, AISTATS 2013

Sampling a matrix Gaussian

Sampling a matrix Gaussian

In order to sample the matrix Gaussian at each layer we have to do:

$$\mathbf{B} = \psi(\mathbf{A})\mathbf{M}_h + \mathbf{K}_{in}(\mathbf{A},\mathbf{A})^{\frac{1}{2}}\mathbf{E}\mathbf{K}_{out}^{\frac{1}{2}}$$

$$\mathbf{E}[i,j] \sim \mathcal{N}(0,1)$$

Sampling a matrix Gaussian

In order to sample the matrix Gaussian at each layer we have to do:

$$\mathbf{B} = \psi(\mathbf{A})\mathbf{M}_h + \mathbf{K}_{in}(\mathbf{A}, \mathbf{A})^{\frac{1}{2}}\mathbf{E}\mathbf{K}_{out}^{\frac{1}{2}}$$

$$\mathbf{E}[i, j] \sim \mathcal{N}(0, 1)$$

Expensive since we have to compute the square root of the input kernel for each mini-batch.

Inducing points for BNNs

Inducing points for BNNs

To scale it up we introduce pseudo input-output pairs and assume conditional independence for the elements of **B** (FITC GP approximation):

$$
\begin{aligned}
\mathbf{b} &\sim p(\mathbf{b}|\mathbf{a}, \tilde{\mathbf{B}}, \tilde{\mathbf{A}}) \\
&\sim \mathcal{N}(\psi(\mathbf{a})\mathbf{M}_h + \mathbf{K}_{in}(\mathbf{a}, \tilde{\mathbf{A}})\mathbf{K}_{in}(\tilde{\mathbf{A}}, \tilde{\mathbf{A}})^{-1}(\tilde{\mathbf{B}} - \tilde{\mathbf{A}}\mathbf{M}_h), \\
&\qquad \mathbf{K}_{out} \odot (\mathbf{K}_{in}(\mathbf{a}, \mathbf{a}) - \mathbf{K}_{in}(\mathbf{a}, \tilde{\mathbf{A}})\mathbf{K}_{in}(\tilde{\mathbf{A}}, \tilde{\mathbf{A}})^{-1}\mathbf{K}_{in}(\mathbf{a}, \tilde{\mathbf{A}})^T))
\end{aligned}
$$

Inducing points for BNNs

To scale it up we introduce pseudo input-output pairs and assume conditional independence for the elements of **B** (FITC GP approximation):

$$\begin{aligned}
\mathbf{b} &\sim p(\mathbf{b}|\mathbf{a}, \tilde{\mathbf{B}}, \tilde{\mathbf{A}}) \\
&\sim \mathcal{N}(\psi(\mathbf{a})\mathbf{M}_h + \mathbf{K}_{in}(\mathbf{a}, \tilde{\mathbf{A}})\mathbf{K}_{in}(\tilde{\mathbf{A}}, \tilde{\mathbf{A}})^{-1}(\tilde{\mathbf{B}} - \tilde{\mathbf{A}}\mathbf{M}_h), \\
&\qquad \mathbf{K}_{out} \odot (\mathbf{K}_{in}(\mathbf{a}, \mathbf{a}) - \mathbf{K}_{in}(\mathbf{a}, \tilde{\mathbf{A}})\mathbf{K}_{in}(\tilde{\mathbf{A}}, \tilde{\mathbf{A}})^{-1}\mathbf{K}_{in}(\mathbf{a}, \tilde{\mathbf{A}})^T))
\end{aligned}$$

Advantages

- Sampling scales cubically with the amount of inducing points
- Variance reduction if real inputs are “similar”

Regression experiments

Regression experiments

	Avg. Test RMSE and Std. Errors				Avg. Test LL and Std. Errors			
Dataset	**VI**	**PBP**	**Dropout**	**VMG**	**VI**	**PBP**	**Dropout**	**VMG**
Boston	4.32±0.29	3.01±0.18	2.97±0.85	**2.81±0.11**	-2.90±0.07	-2.57± 0.09	**-2.46±0.25**	-2.54±0.08
Concrete	7.19±0.12	5.67±0.09	5.23± 0.53	**4.70±0.14**	-3.39±0.02	-3.16±0.02	-3.04±0.09	**-2.98±0.03**
Energy	2.65±0.08	1.80±0.05	1.66±0.19	**1.16±0.03**	-2.39±0.03	-2.04±0.02	-1.99±0.09	**-1.45±0.03**
Kin8nm	0.10±0.00	0.10±0.00	0.10±0.00	**0.08±0.00**	0.90±0.01	0.90±0.01	0.95±0.03	**1.14±0.01**
Naval	0.01±0.00	0.01±0.00	0.01±0.00	**0.00±0.00**	3.73±0.12	3.73±0.01	3.80±0.05	**5.84±0.00**
Pow. Plant	4.33±0.04	4.12±0.03	4.02±0.18	**3.88±0.03**	-2.89±0.01	-2.84±0.01	-2.80±0.05	**-2.78±0.01**
Protein	4.84±0.03	4.73±0.01	4.36±0.04	**4.14±0.01**	-2.99±0.01	-2.97±0.00	-2.89±0.01	**-2.84±0.00**
Wine	0.65±0.01	0.64±0.01	0.62±0.04	**0.61±0.01**	-0.98±0.01	-0.97±0.01	-0.93±0.06	**-0.93±0.02**
Yacht	6.89±0.67	1.02±0.05	1.11±0.38	**0.77±0.06**	-3.43±0.16	-1.63±0.02	-1.55±0.12	**-1.29±0.02**
Year	9.034±NA	8.879±NA	8.849±NA	**8.780±NA**	-3.622±NA	-3.603±NA	**-3.588±NA**	-3.589±NA

Table 1. Average test set RMSE, predictive log-likelihood and standard errors for the regression datasets. VI, PBP and Dropout correspond to the variational inference method of (Graves, 2011), probabilistic backpropagation (Hernández-Lobato & Adams, 2015) and dropout uncertainty (Gal & Ghahramani, 2015). VMG (Variational Matrix Gaussian) corresponds to the proposed model.

Classification experiments

Classification experiments

Method	# layers	Test err.
Max. Likel. (Simard et al., 2003)	2×800	1.60
Dropout (Srivastava, 2013)	-	1.25
DropConnect (Wan et al., 2013)	2×800	1.20
Bayes B. SM (Blundell et al., 2015)	2×400	1.36
	2×800	1.34
	2×1200	1.32
Var. Dropout (Kingma et al., 2015)	3×150	≈ 1.42
	3×250	≈ 1.28
	3×500	≈ 1.18
	3×750	≈ 1.09
VMG	2×400	**1.15**
	3×150	1.18
	3×250	1.11
	3×500	1.08
	3×750	**1.05**

Table 2. Test errors for the permutation invariant MNIST dataset. Bayes B. SM correspond to Bayes by Backprop with the scale mixture prior and the variational dropout results are from the Variational (A) model that doesn't downscale the KL-divergence (so as to keep the comparison fair).

Part 2

Model uncertainty from BNNs in classification

Opinion: Why are we Bayesian?

Opinion: Why are we Bayesian?

- Combatting overfitting should not be the primary reason for a Bayesian model!

Opinion: Why are we Bayesian?

- Combatting overfitting should not be the primary reason for a Bayesian model!
- Being Bayesian is all about handling the uncertainty

Opinion: Why are we Bayesian?

- Combatting overfitting should not be the primary reason for a Bayesian model!
- Being Bayesian is all about handling the uncertainty
  - We can translate parameter uncertainty into *decision* uncertainty

Opinion: Why are we Bayesian?

- Combatting overfitting should not be the primary reason for a Bayesian model!
- Being Bayesian is all about handling the uncertainty
    - We can translate parameter uncertainty into *decision* uncertainty
- Model uncertainty is crucial for applications that involve decisions

Opinion: Why are we Bayesian?

- Combatting overfitting should not be the primary reason for a Bayesian model!
- Being Bayesian is all about handling the uncertainty
  - We can translate parameter uncertainty into *decision* uncertainty
- Model uncertainty is crucial for applications that involve decisions
  - Self-driving cars

Opinion: Why are we Bayesian?

- Combatting overfitting should not be the primary reason for a Bayesian model!
- Being Bayesian is all about handling the uncertainty
  - We can translate parameter uncertainty into *decision* uncertainty
- Model uncertainty is crucial for applications that involve decisions
  - Self-driving cars
  - Medical applications

How well do deep BNNs capture uncertainty?

- A simple way to measure the quality
  - Visualize class distribution on perturbed unseen inputs[3]
    - Rotations, translations, scalings
  - Ideally the model should be uncertain
- Experiment with Weight decay, Dropout, Matrix Gaussian posteriors, SGLD, Bootstrap

[3] “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”,Yarin Gal, Zoubin Ghahramani, ICML 2016

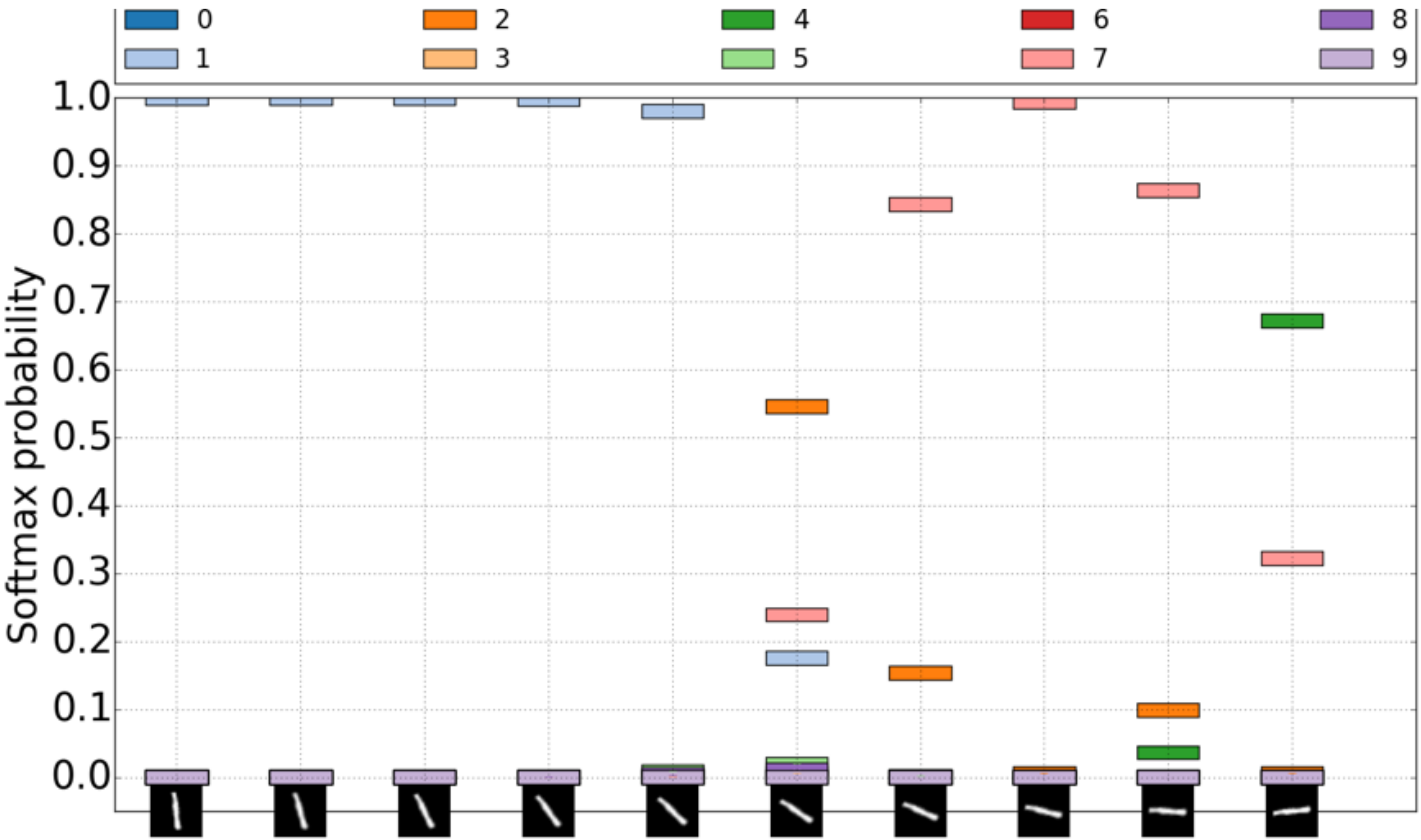
Uncertainty testbed

- Simple LeNet network on MNIST
  - Two convolutional, two fully connected layers
- Dropout network: 0.5 dropout rate for all layers[4]
- Matrix Gaussian network
  - 20 pseudo patches/inputs for convolutional, fully connected layers respectively
- 100 samples for SGLD and 20 for Bootstrap
- All networks (except Bootstrap) had N(0, 1) priors over the parameters

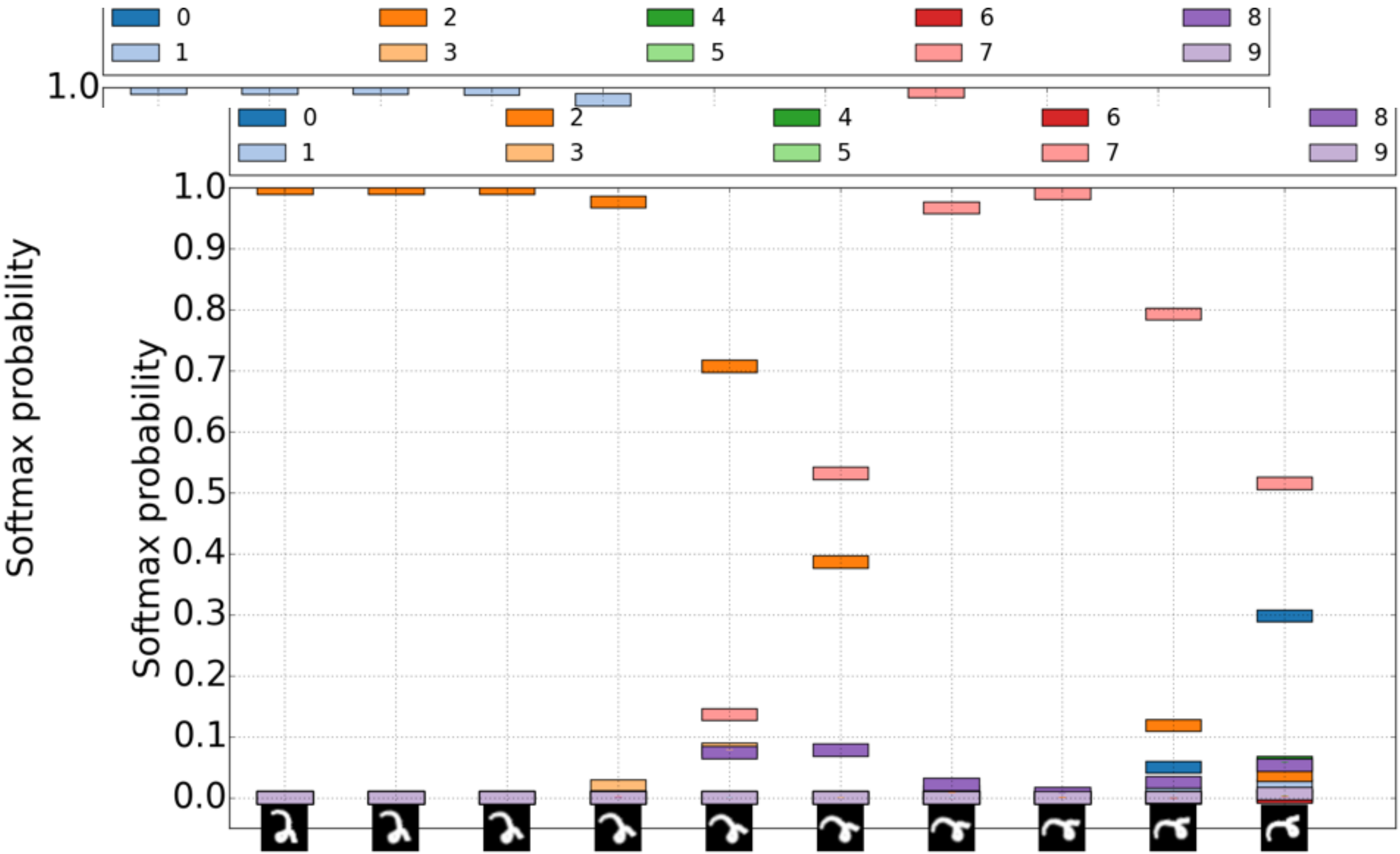
[4] “Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference”,Yarin Gal, Zoubin Ghahramani

Weight decay rotations

Weight decay rotations

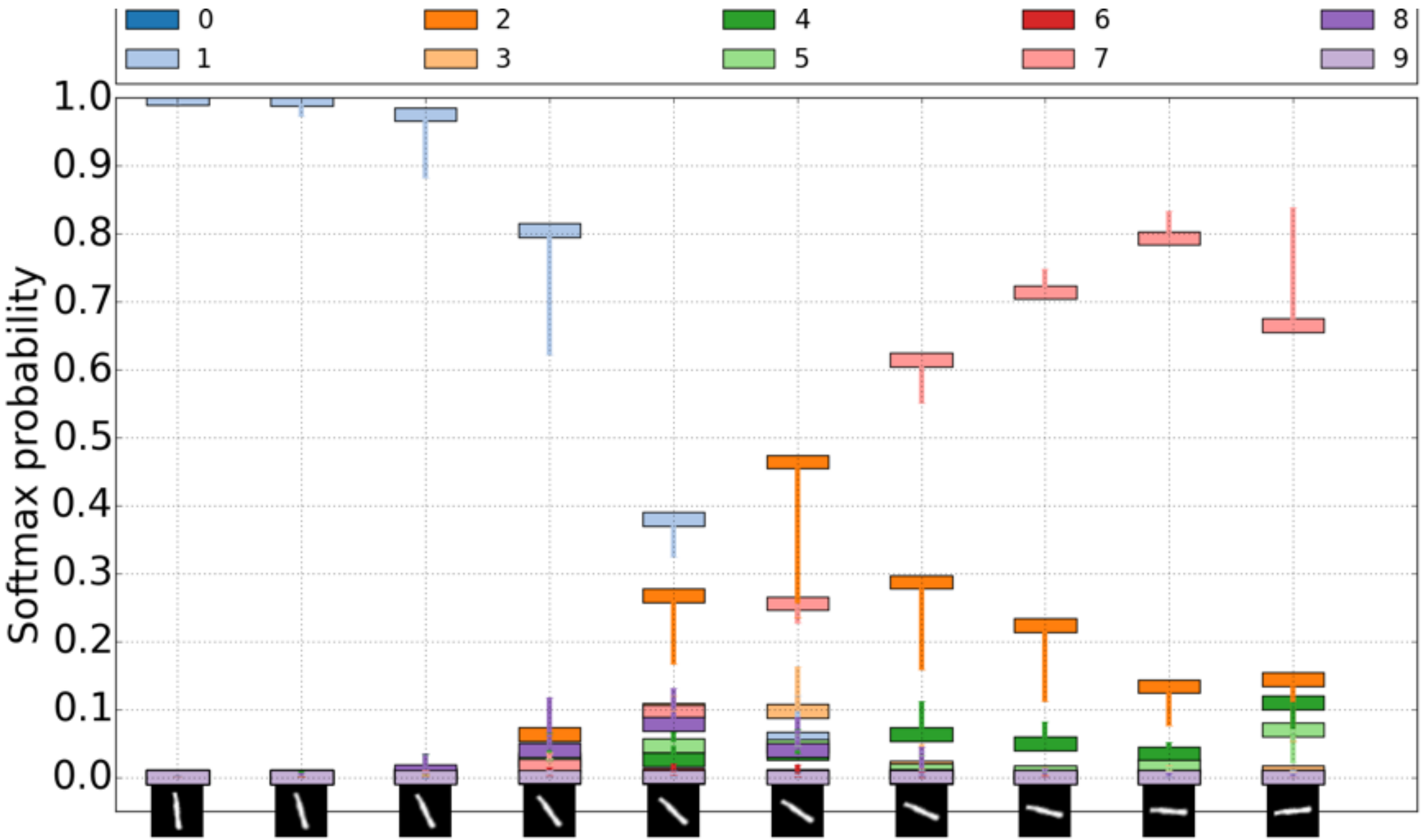


Weight decay rotations

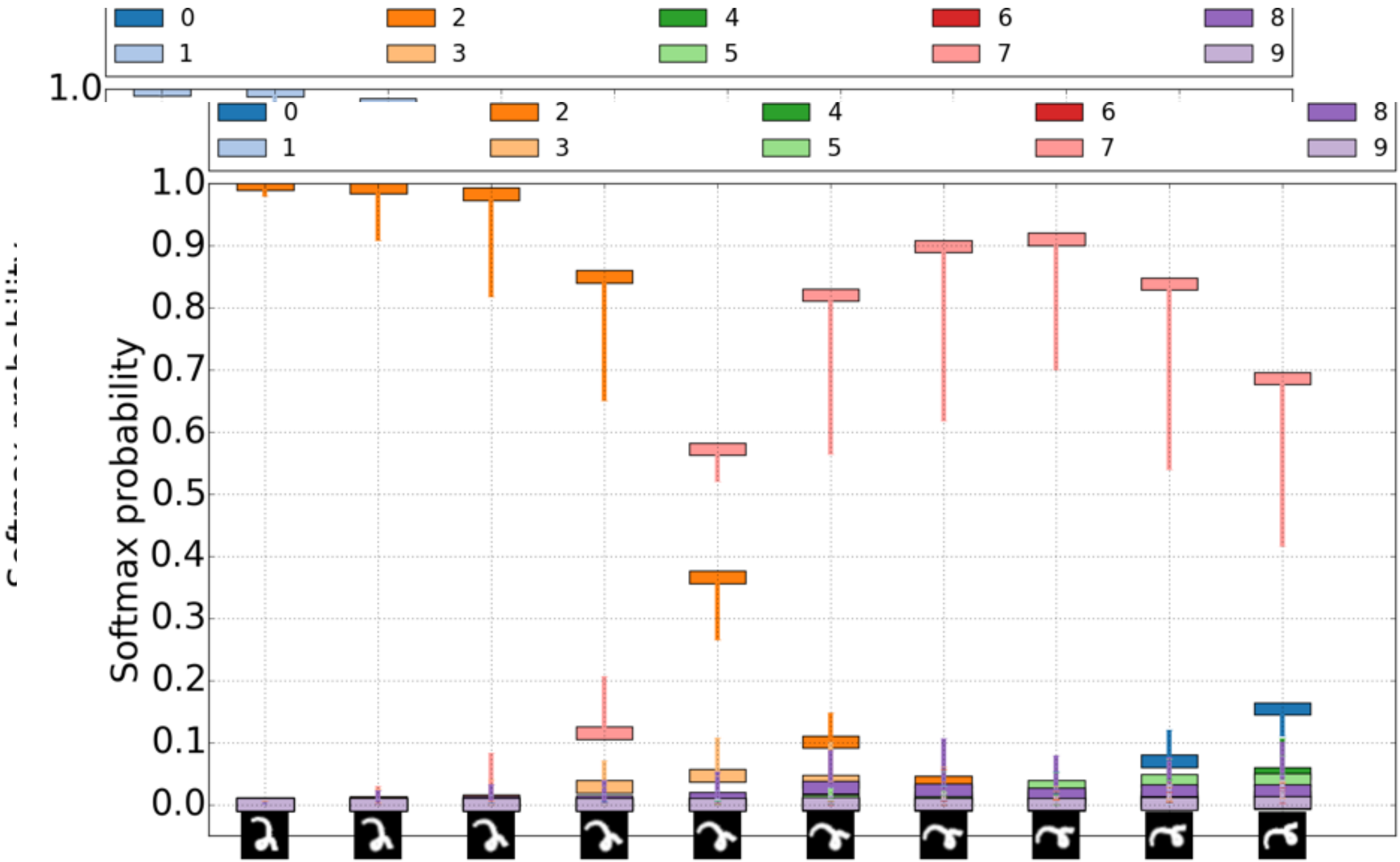


Dropout rotations

Dropout rotations

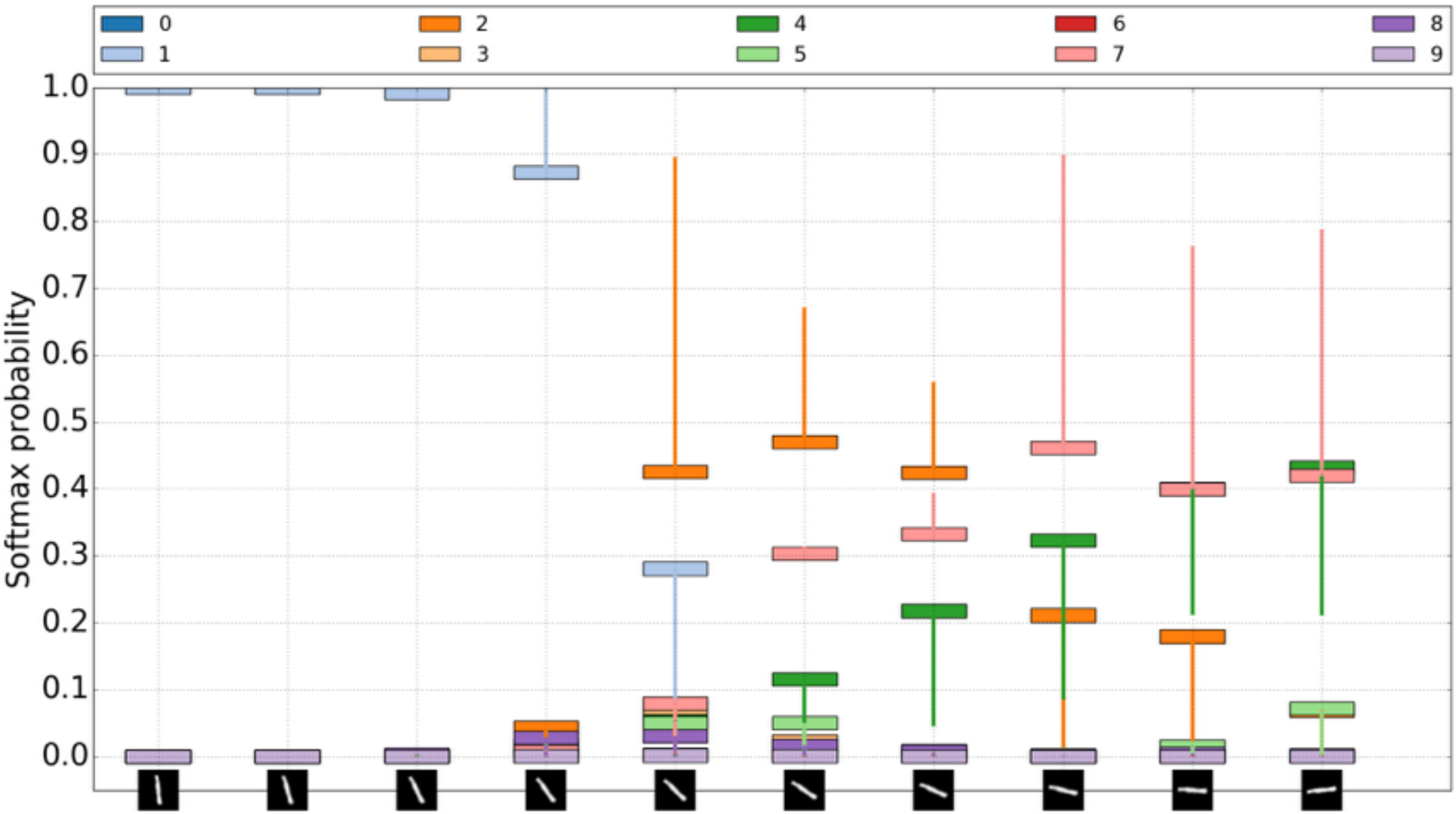


Dropout rotations

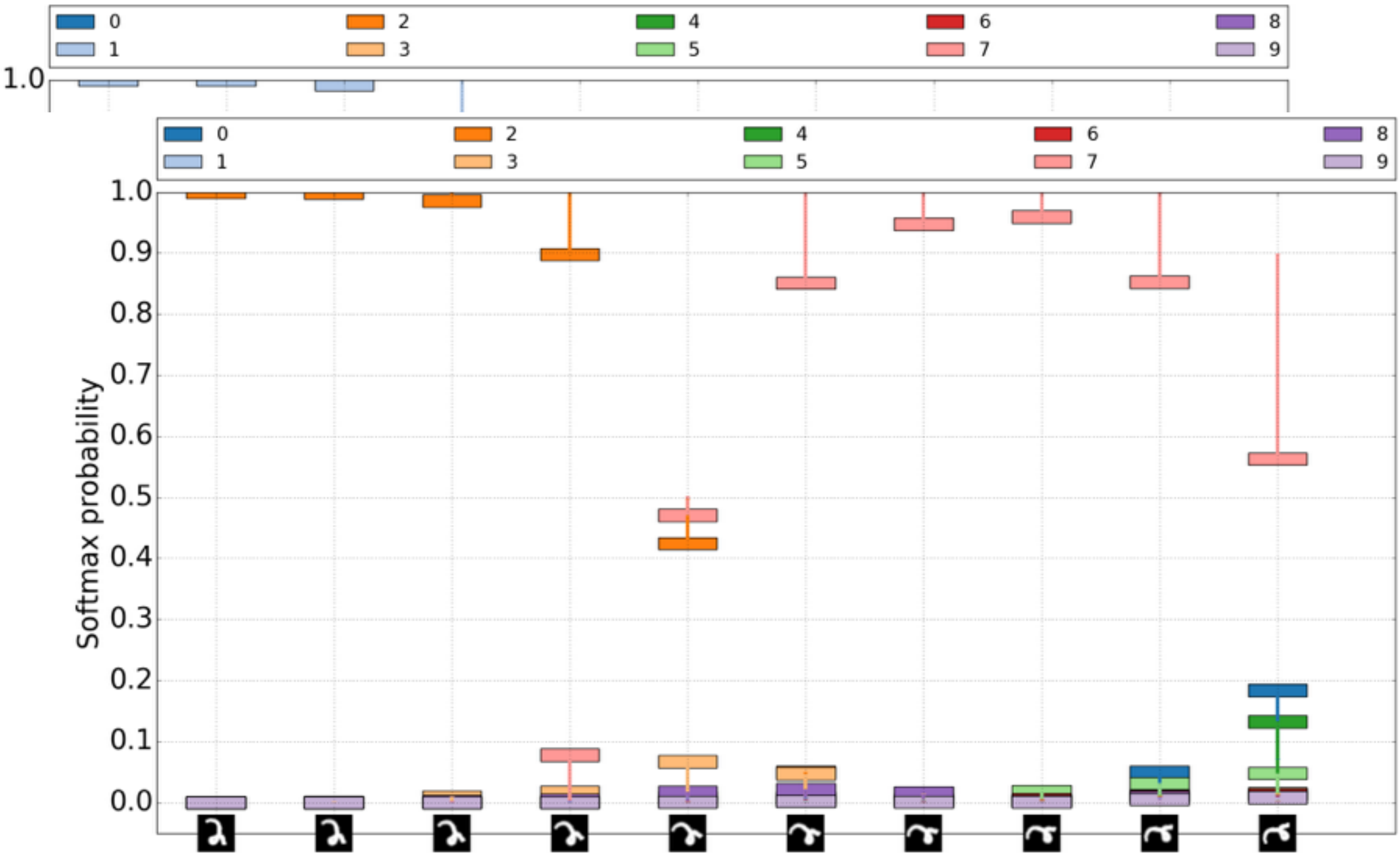


Matrix Gaussian rotations

Matrix Gaussian rotations

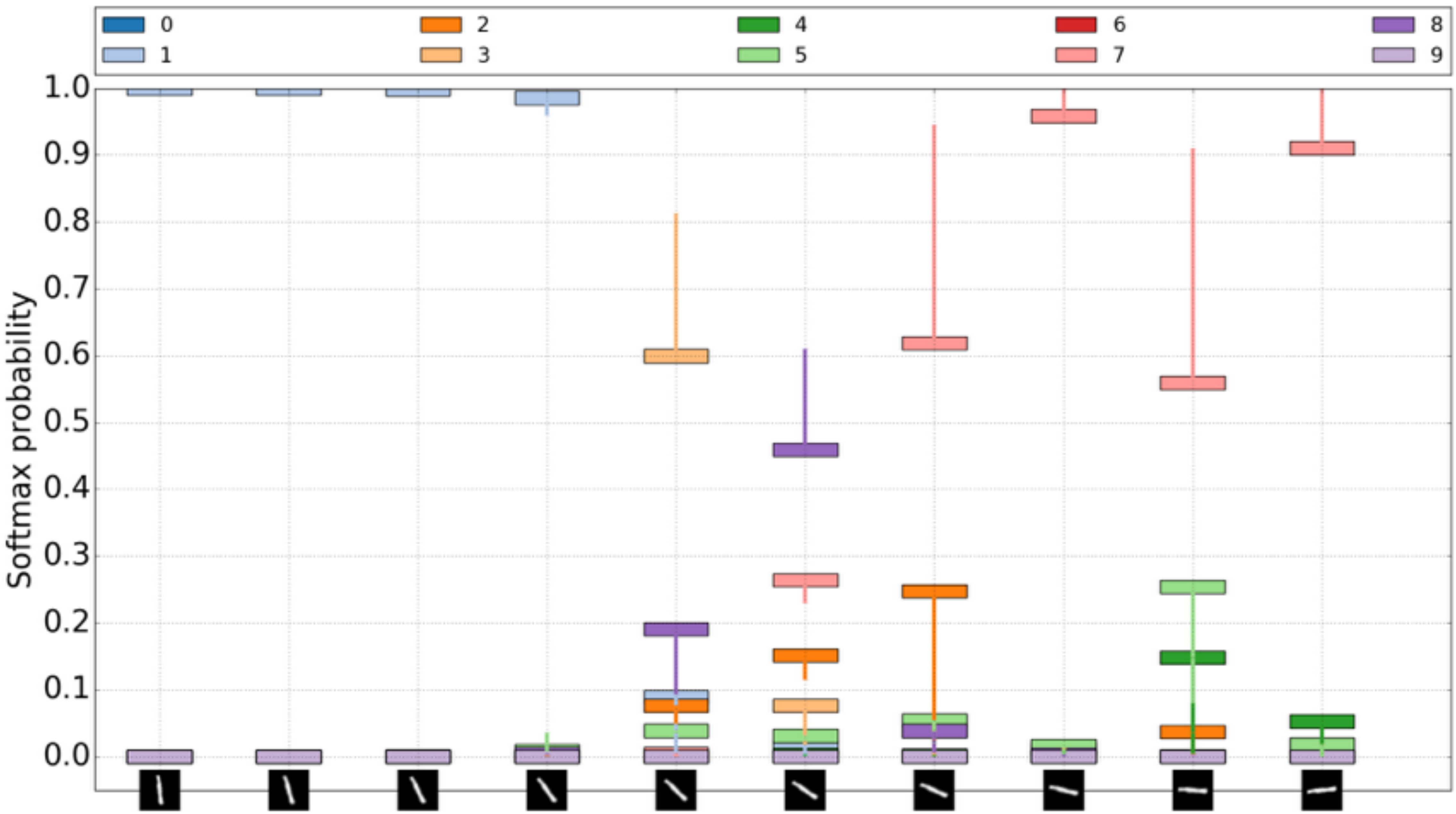


Matrix Gaussian rotations



SGLD rotations

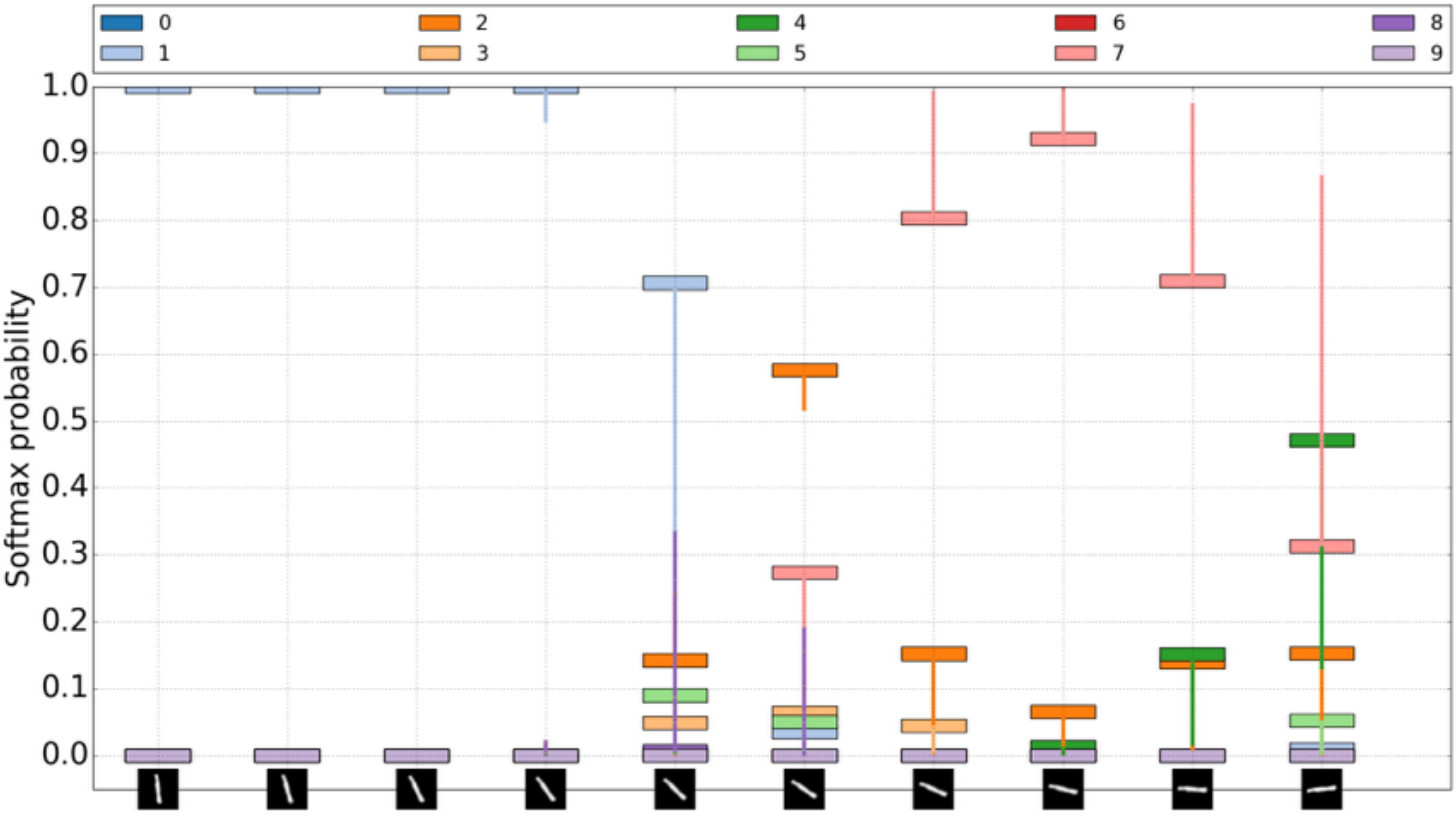
SGLD rotations



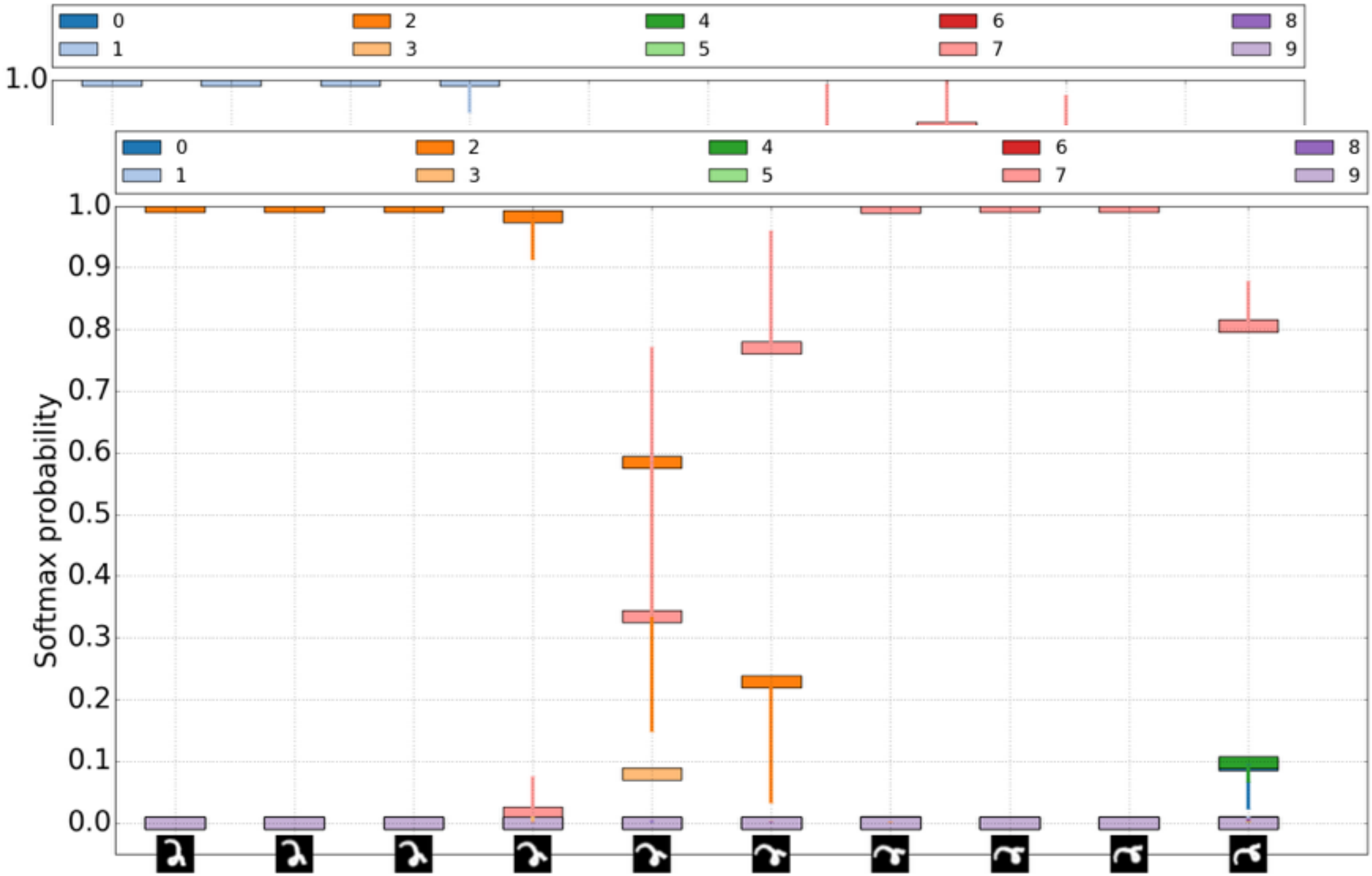
SGLD rotations

Bootstrap rotations

Bootstrap rotations

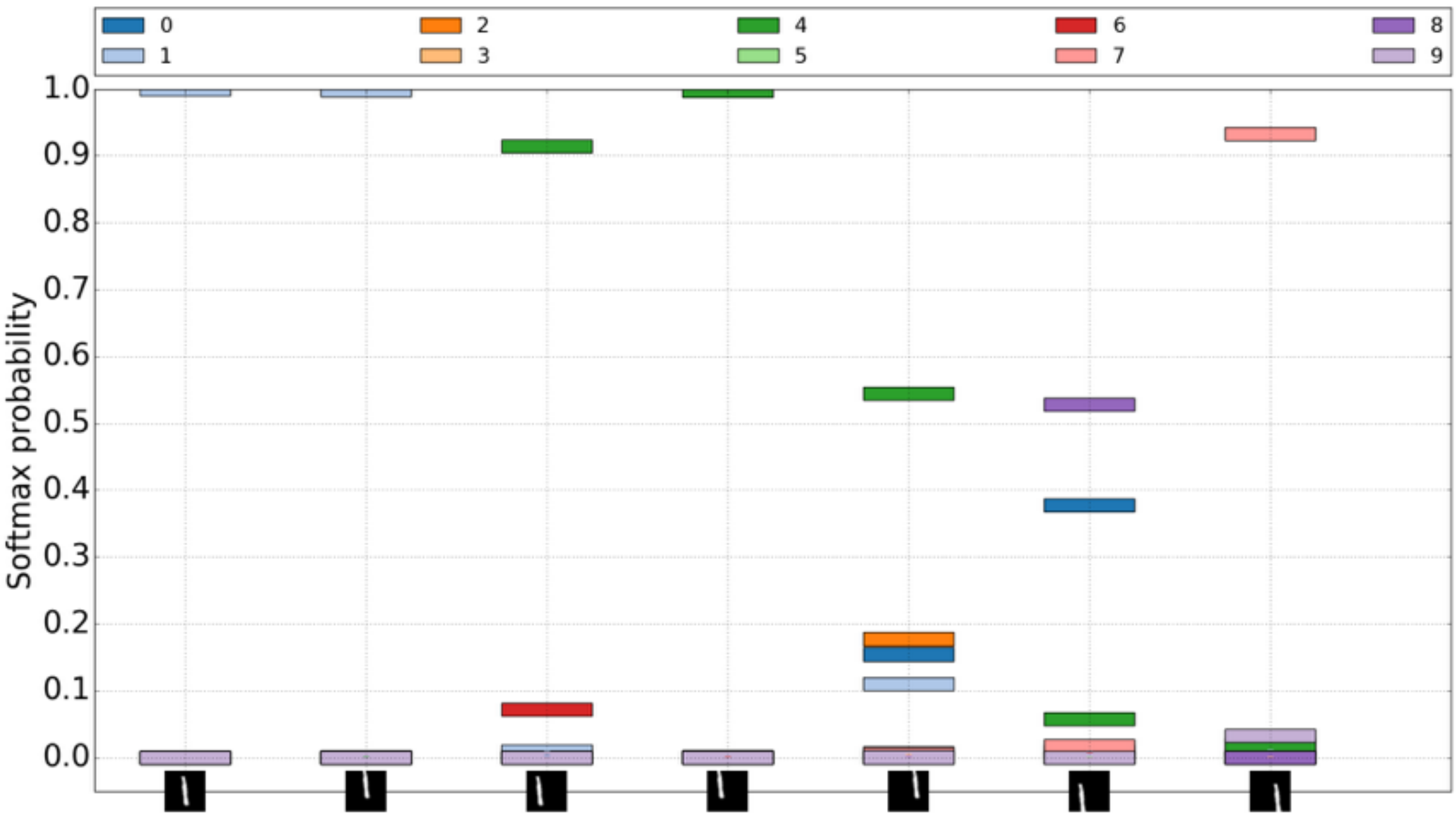


Bootstrap rotations



Weight decay translations

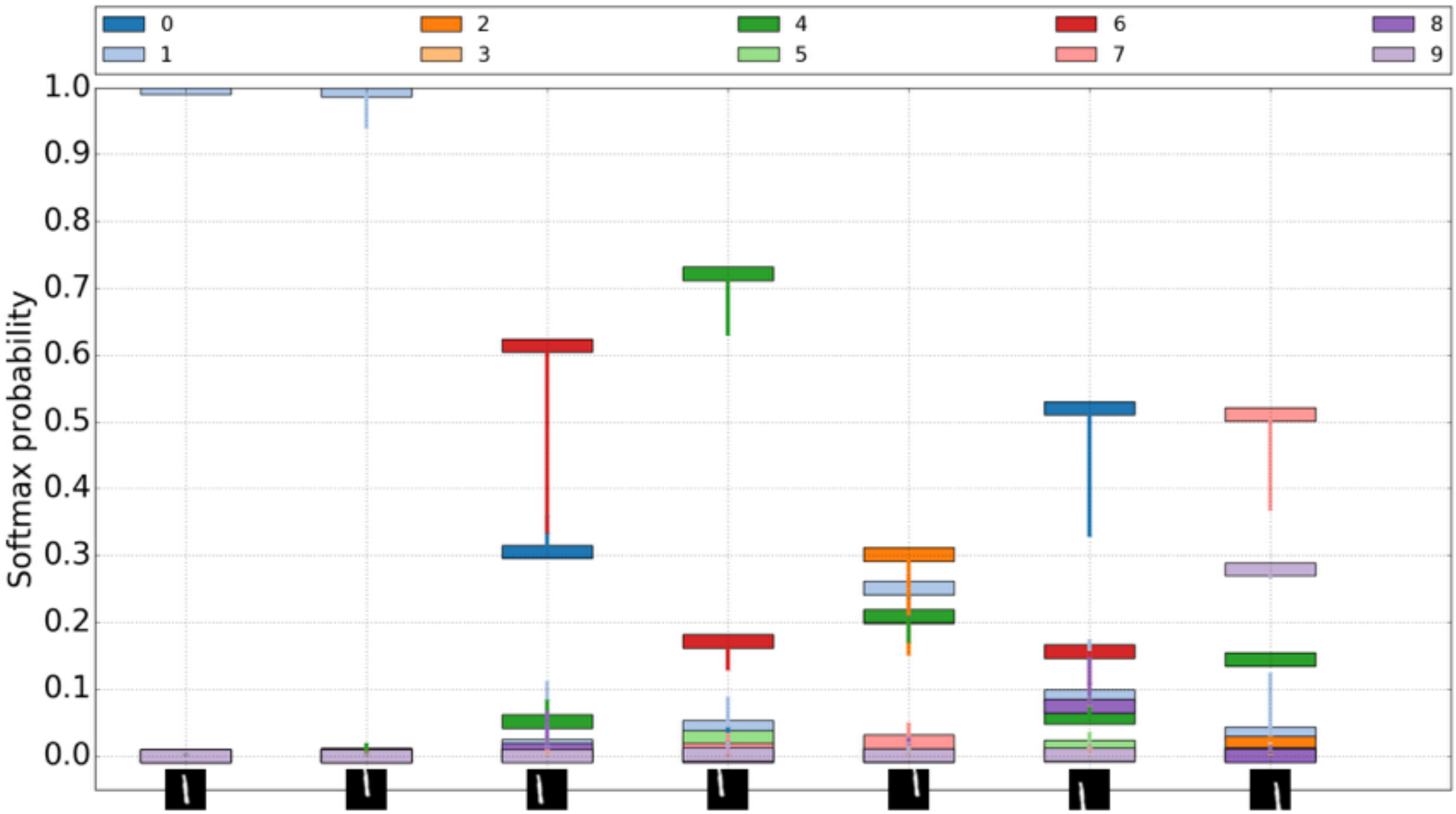
Weight decay translations



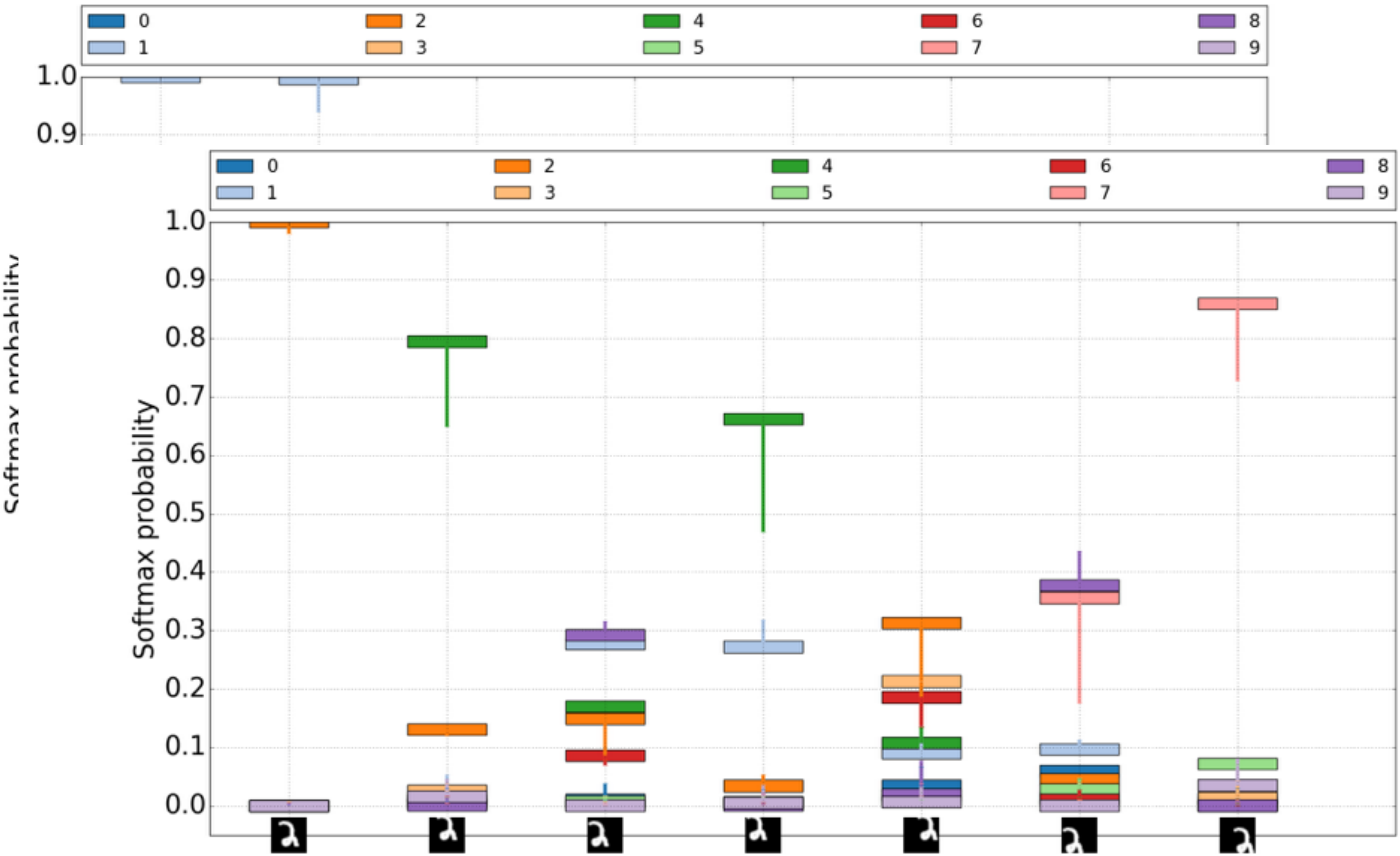
Weight decay translations

Dropout translations

Dropout translations

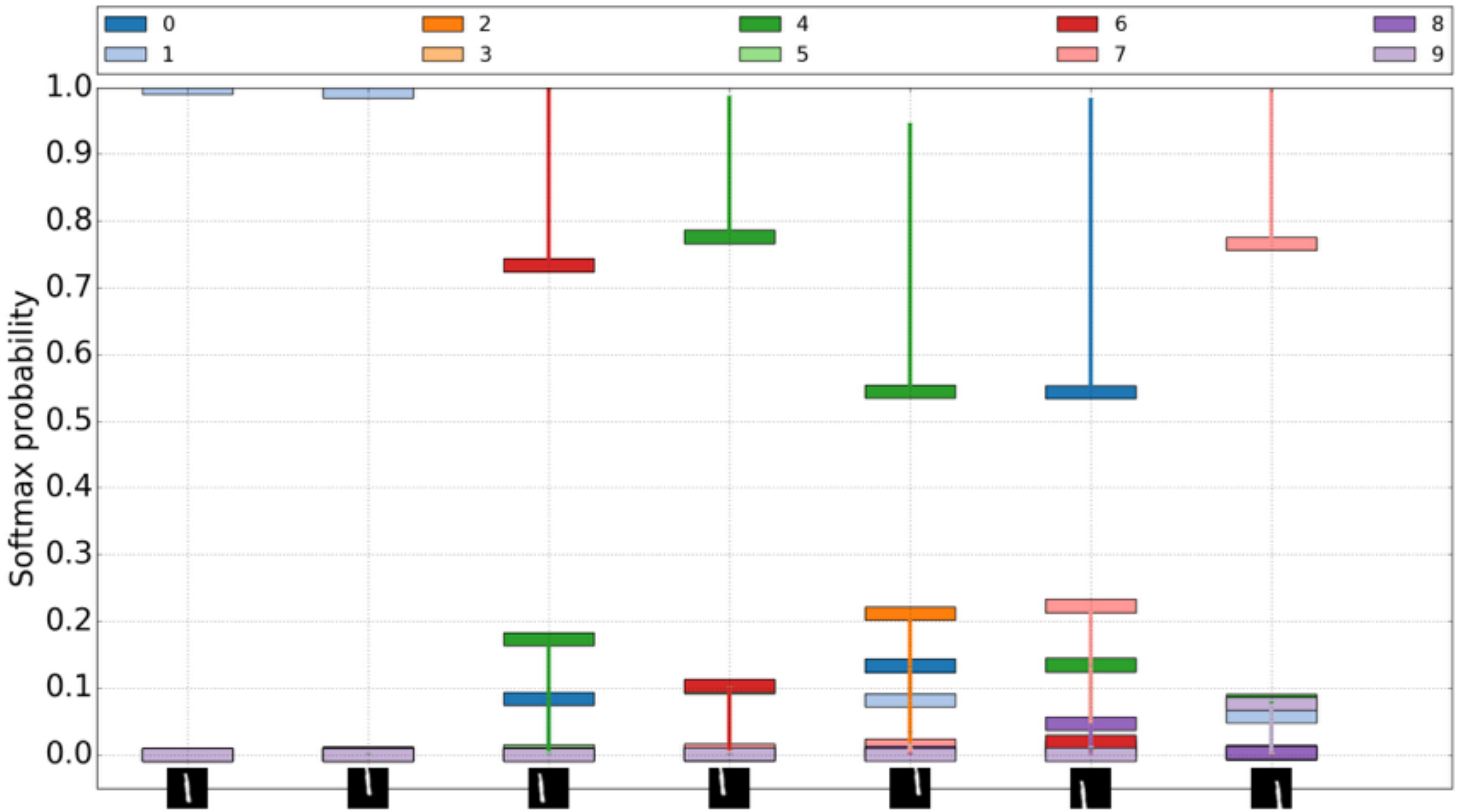


Dropout translations

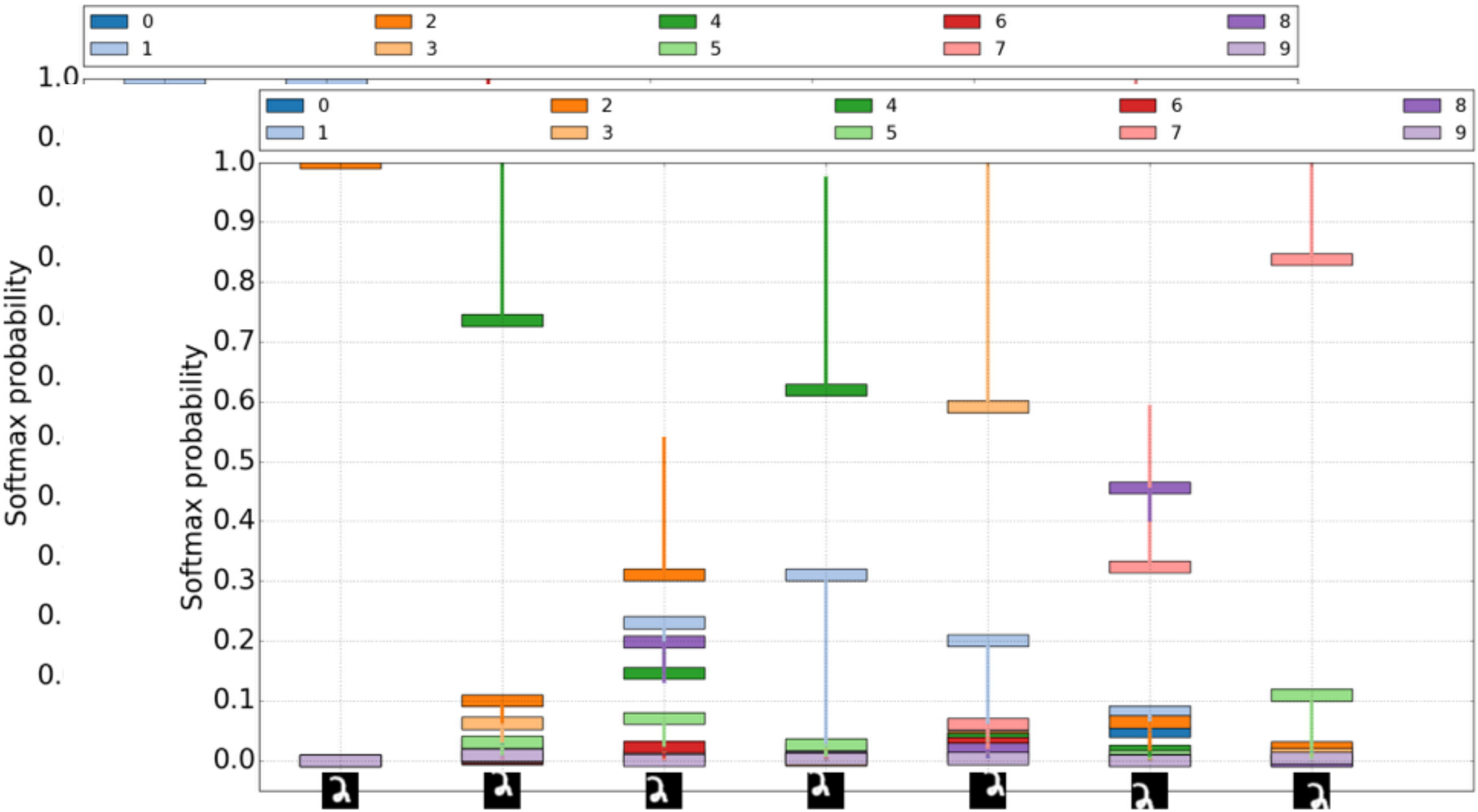


Matrix Gaussian translations

Matrix Gaussian translations

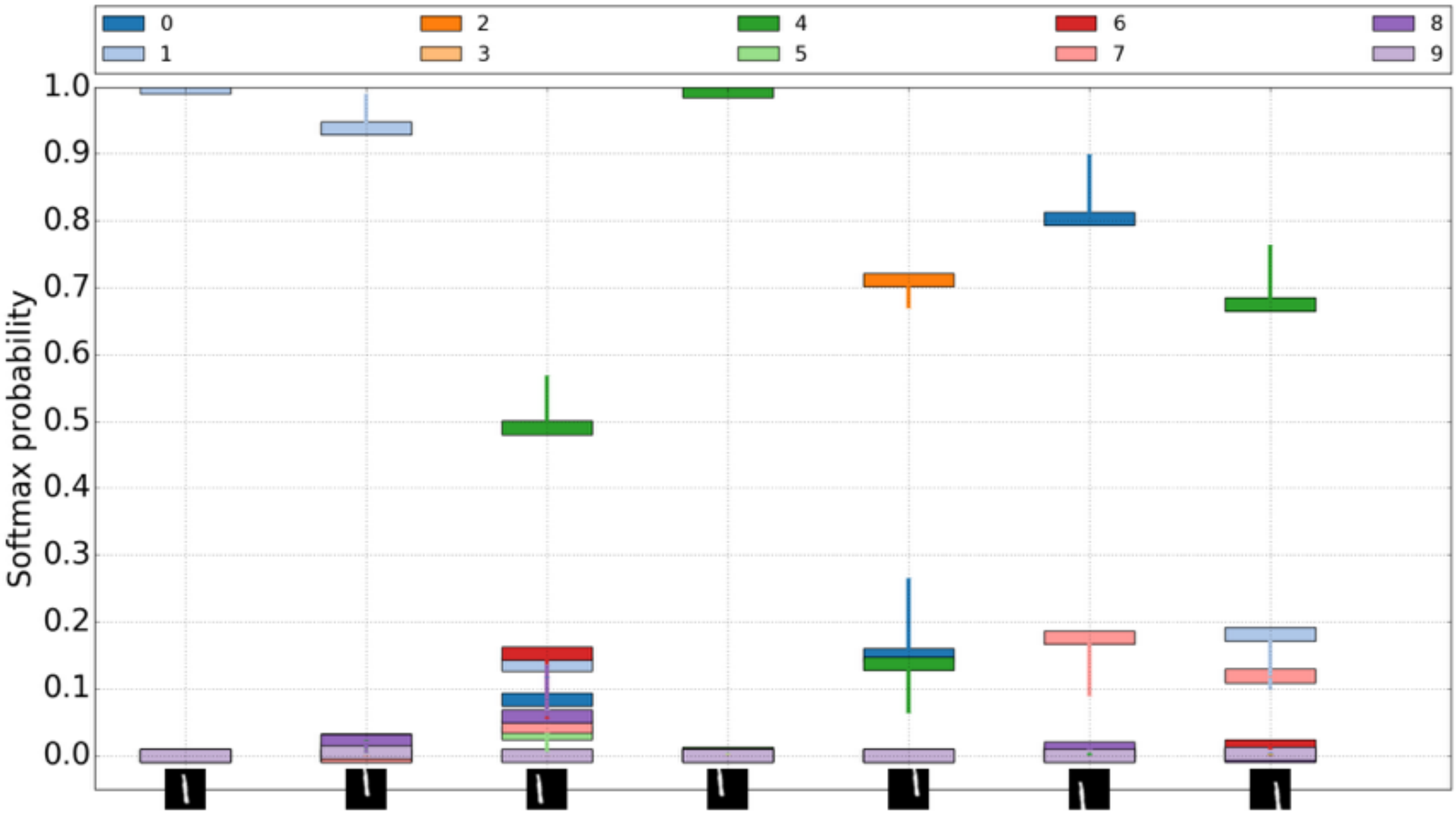


Matrix Gaussian translations

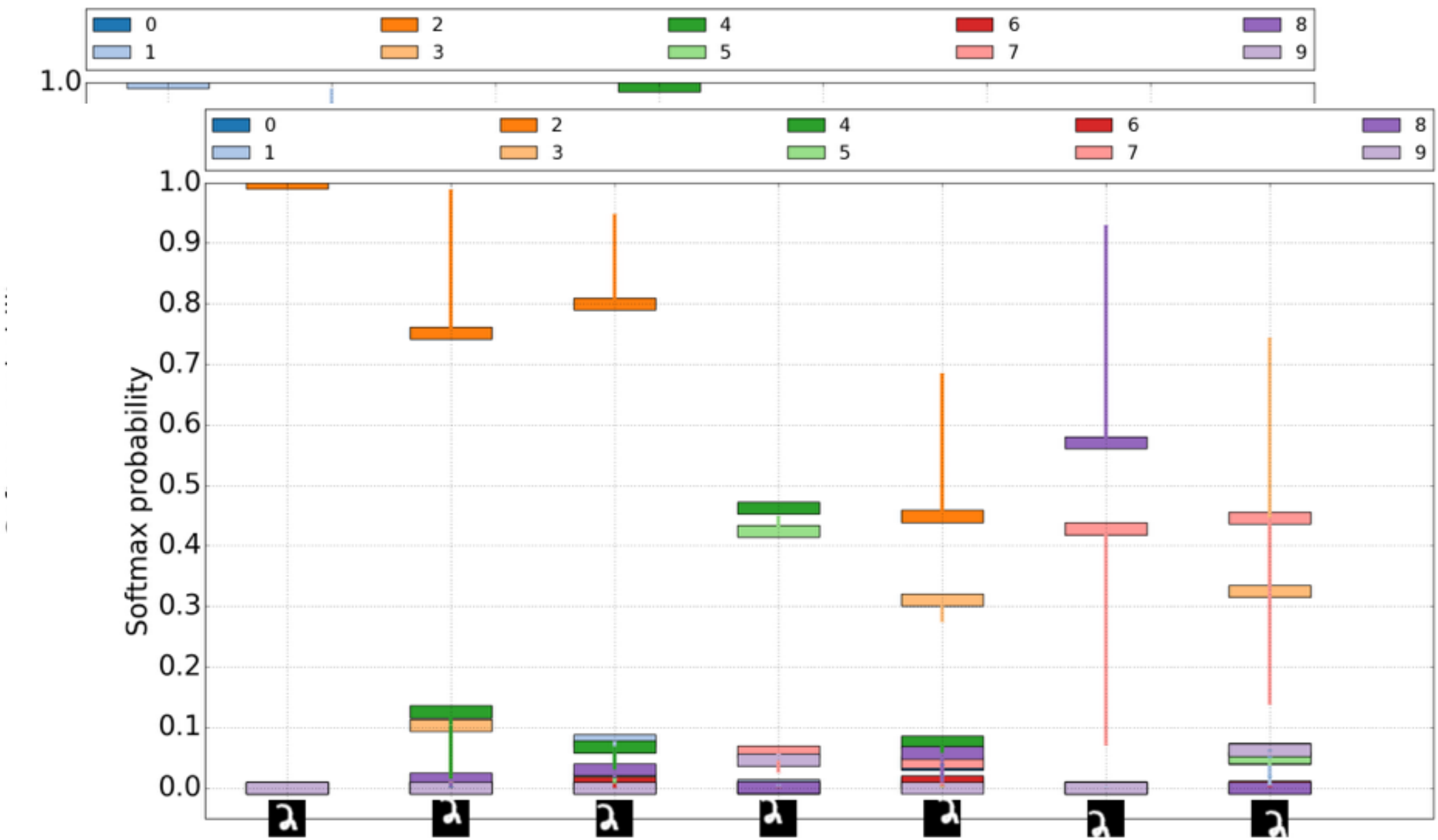


SGLD translations

SGLD translations

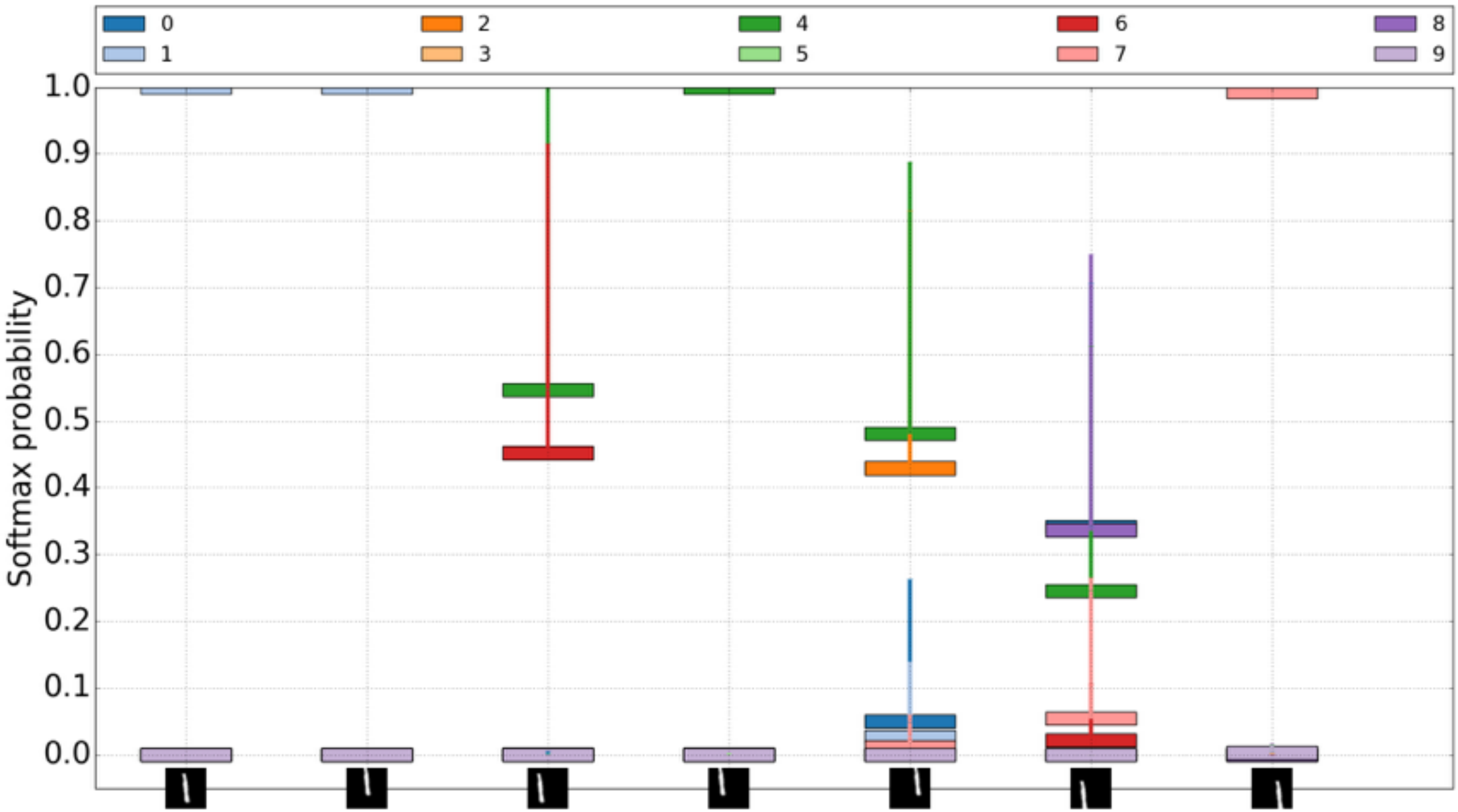


SGLD translations

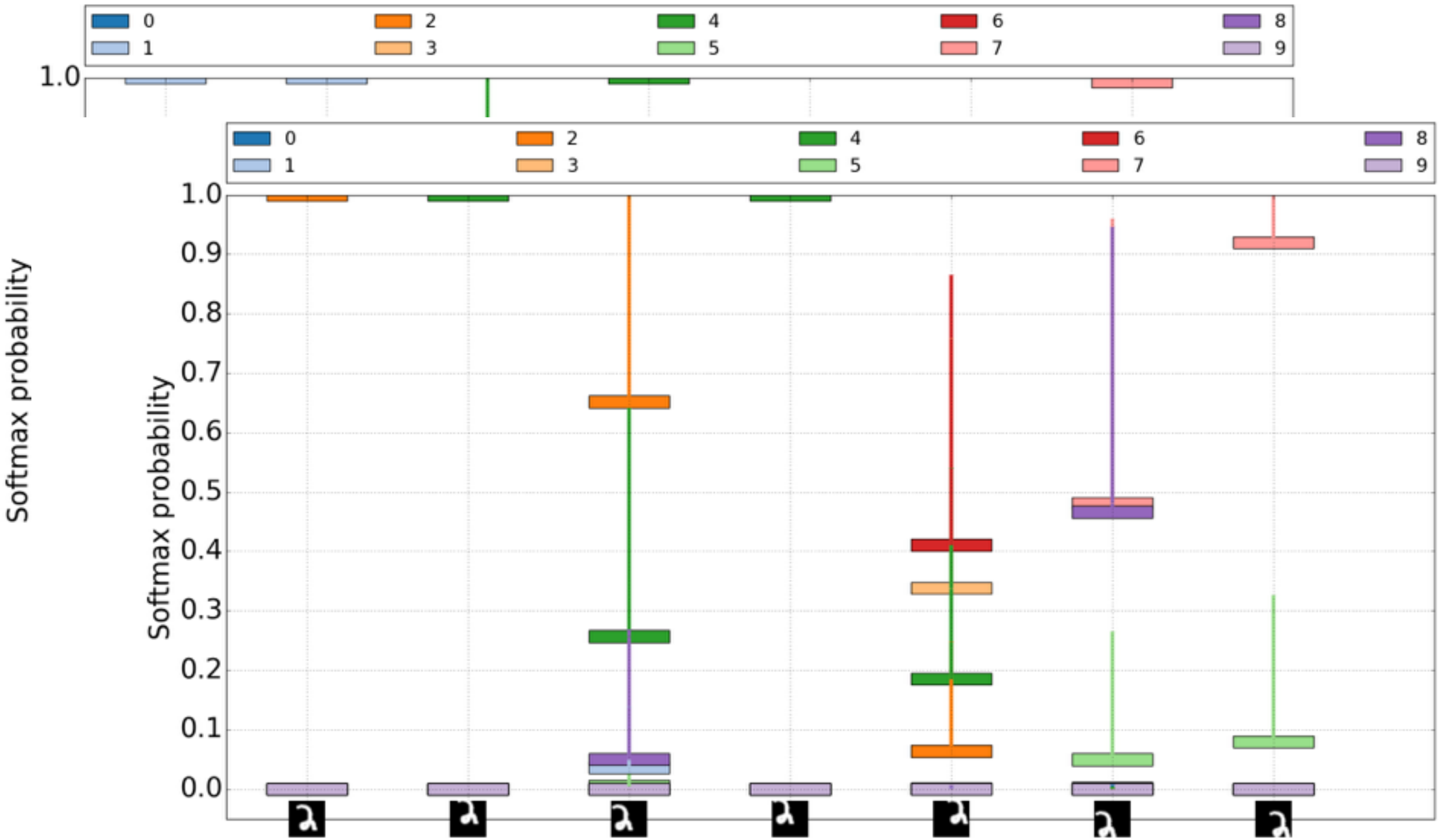


Bootstrap translations

Bootstrap translations

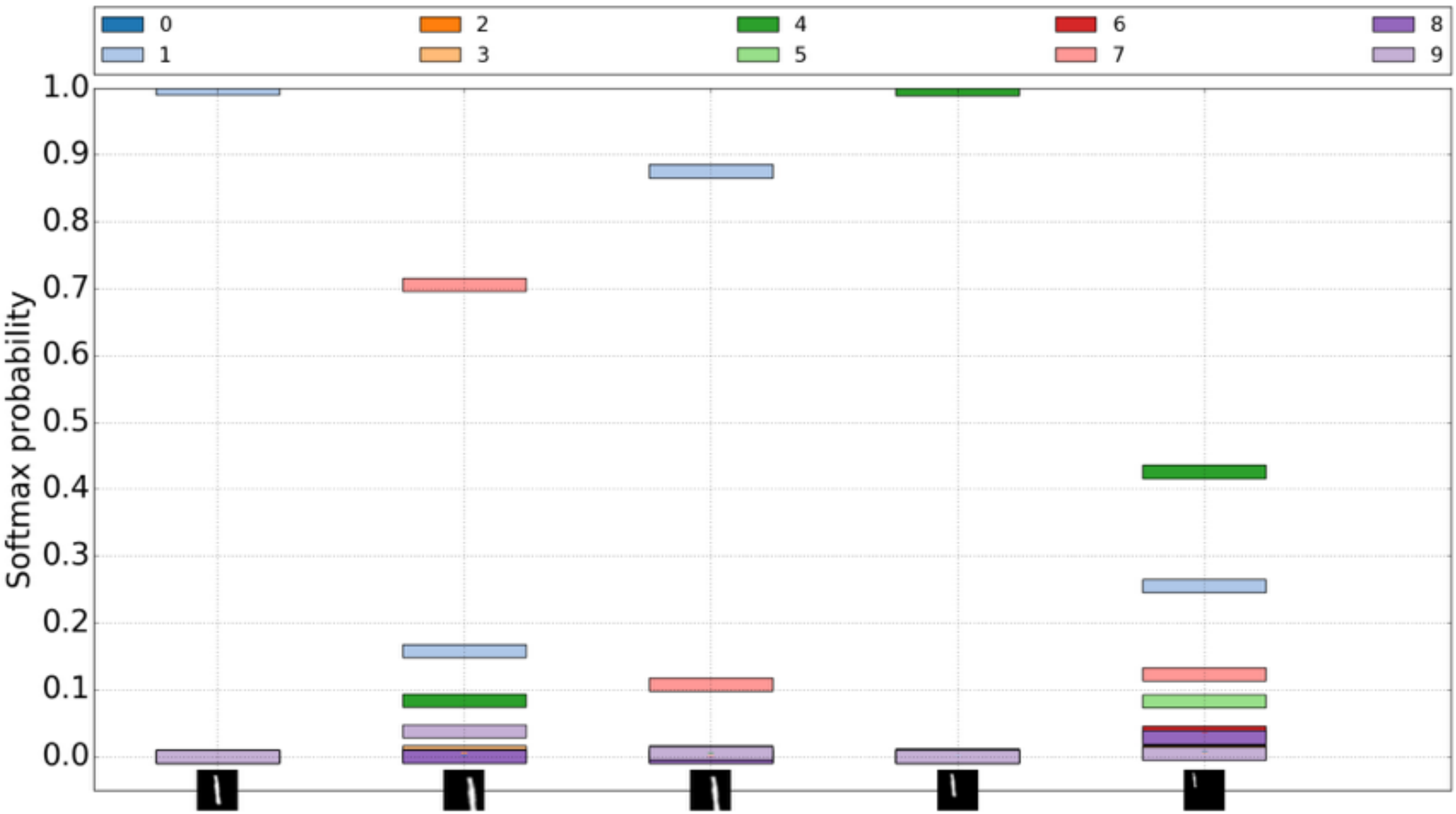


Bootstrap translations

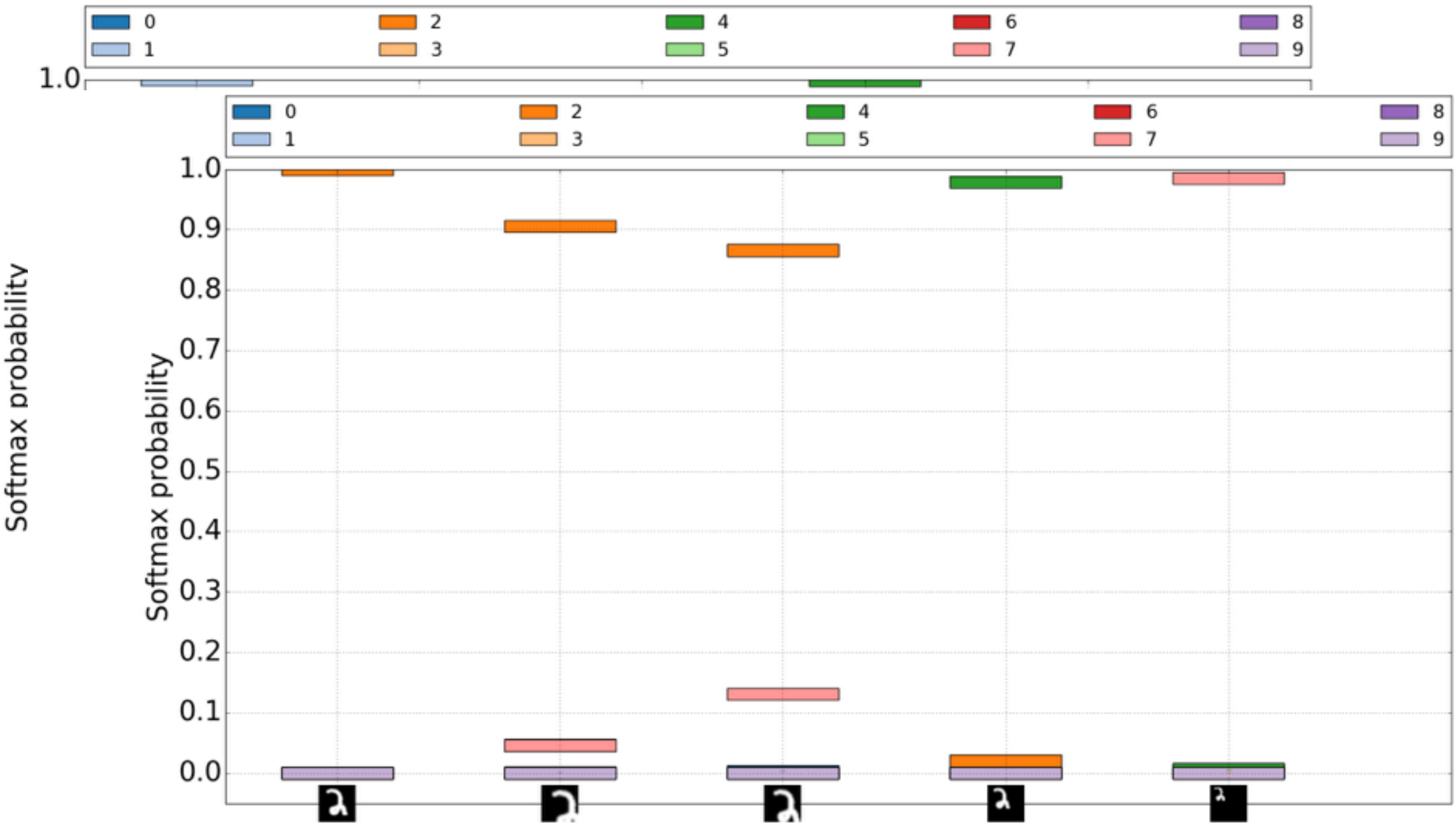


Weight decay scaling

Weight decay scaling

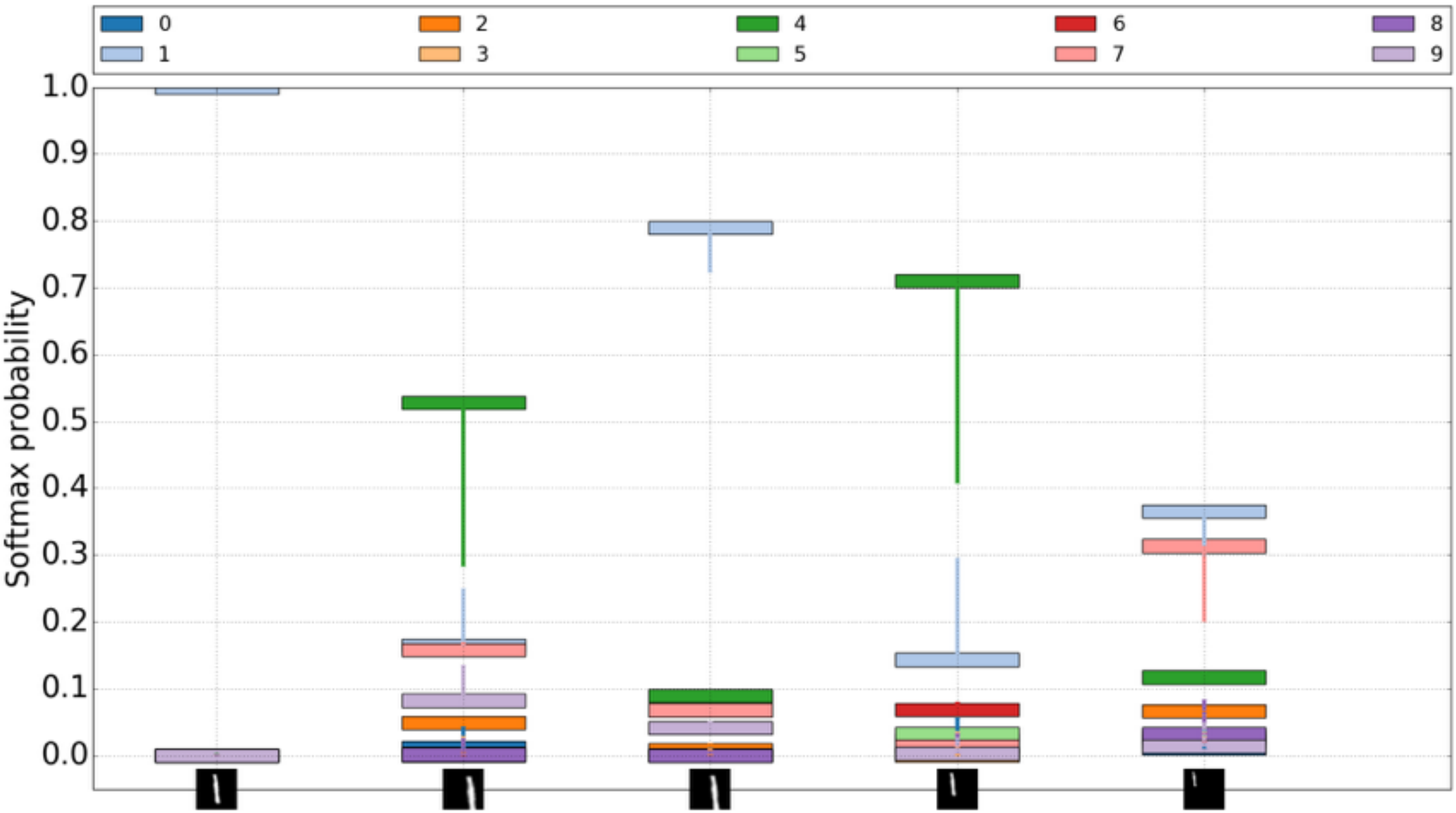


Weight decay scaling

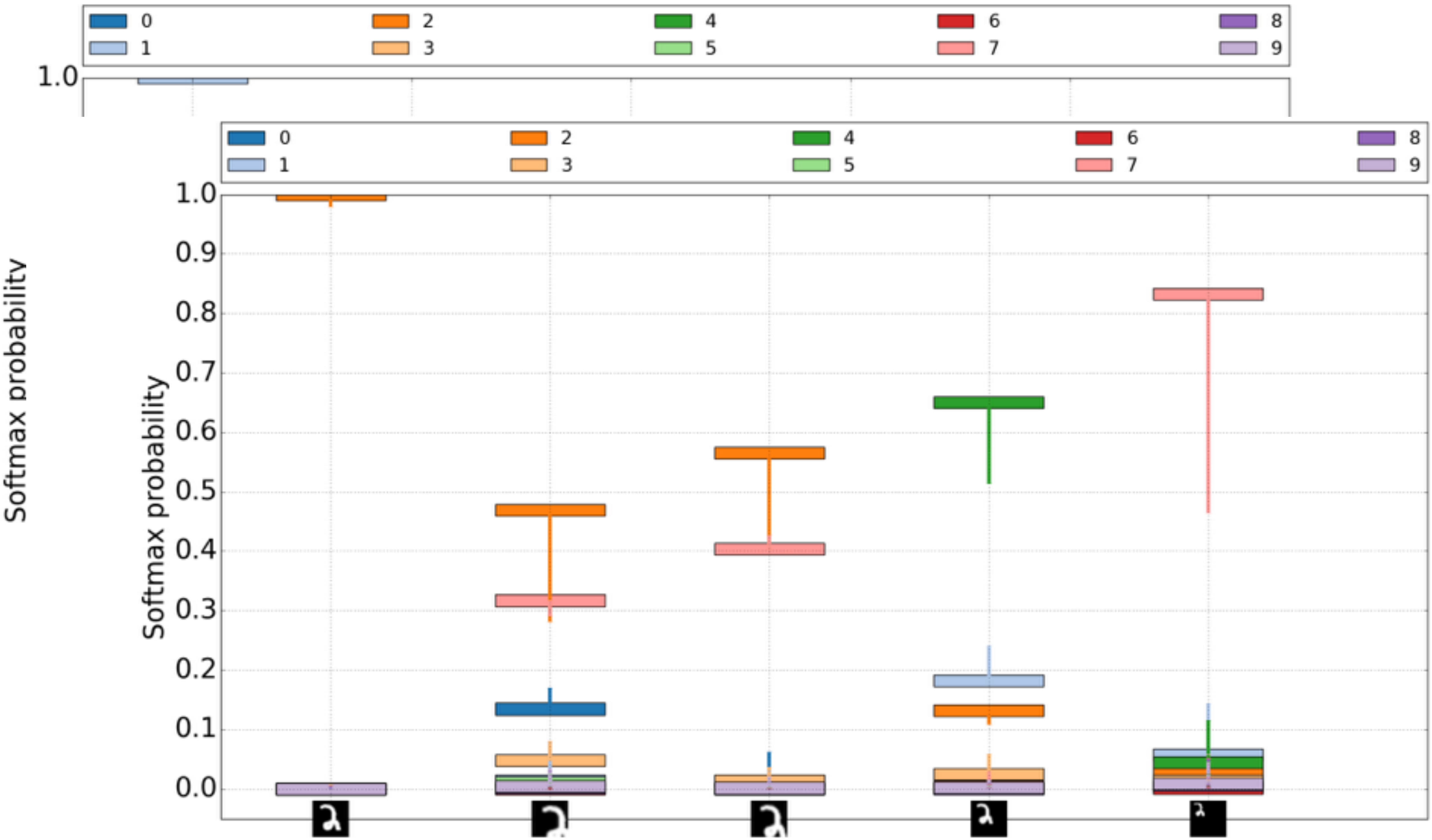


Dropout scaling

Dropout scaling

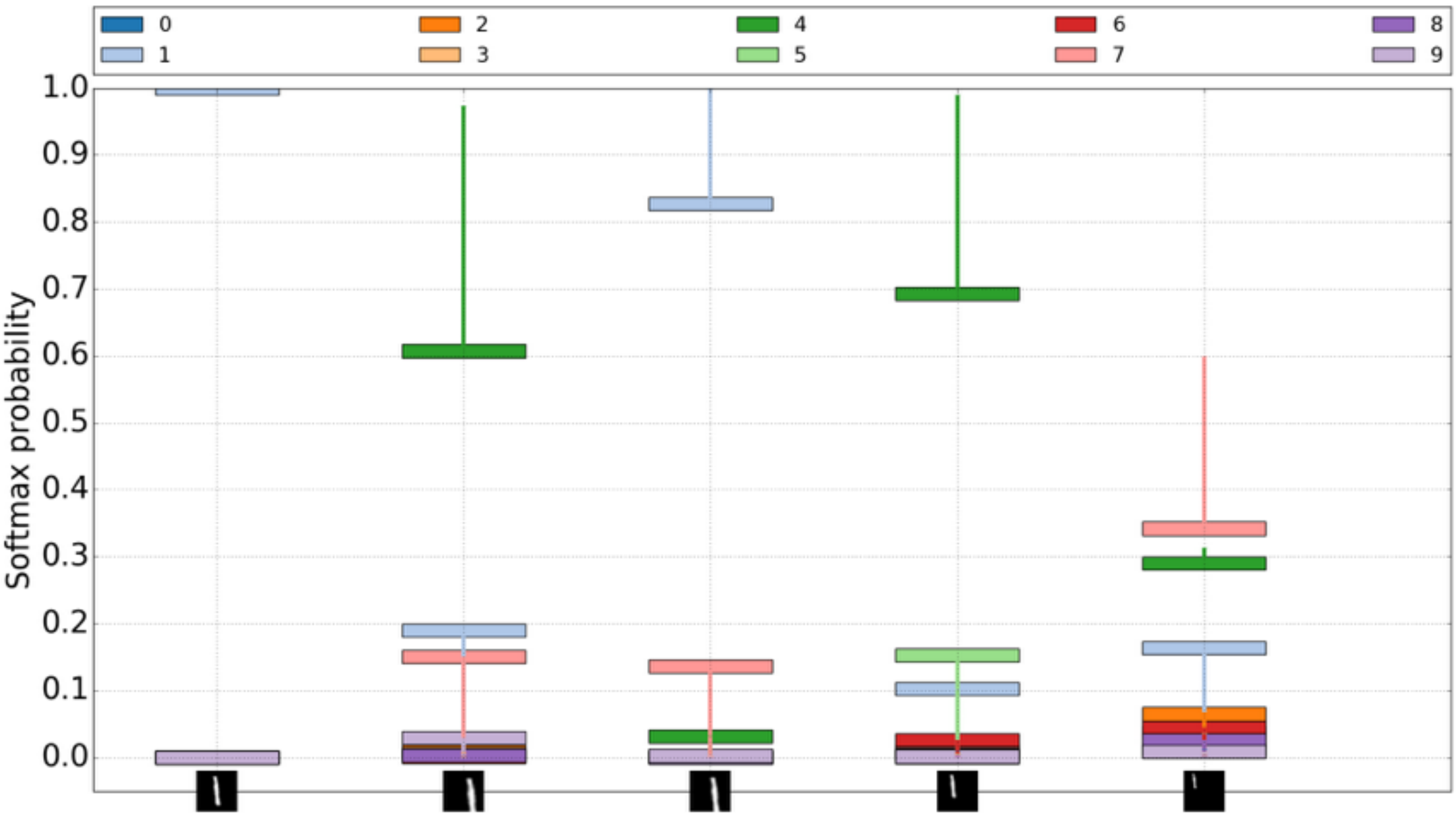


Dropout scaling

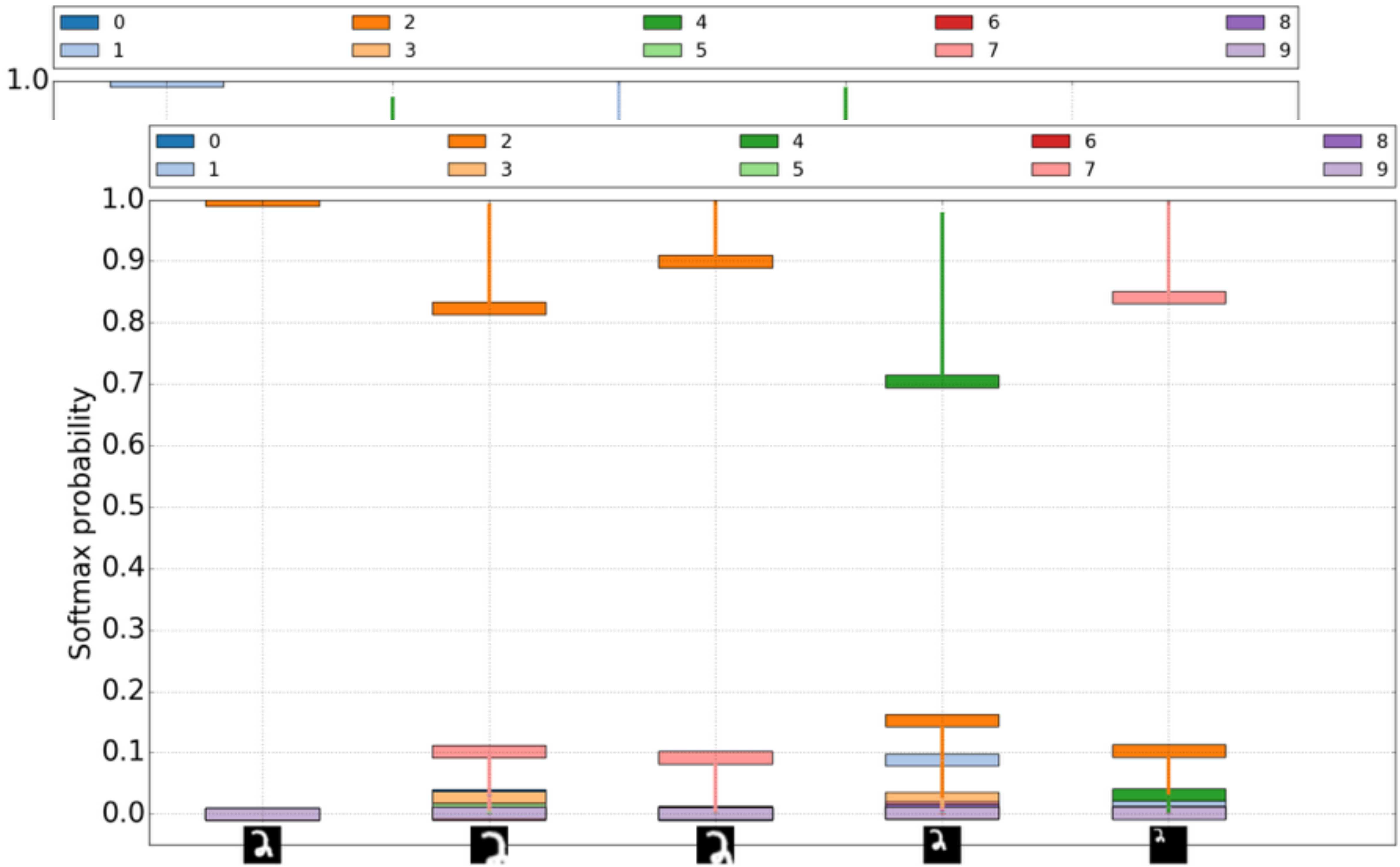


Matrix Gaussian scaling

Matrix Gaussian scaling

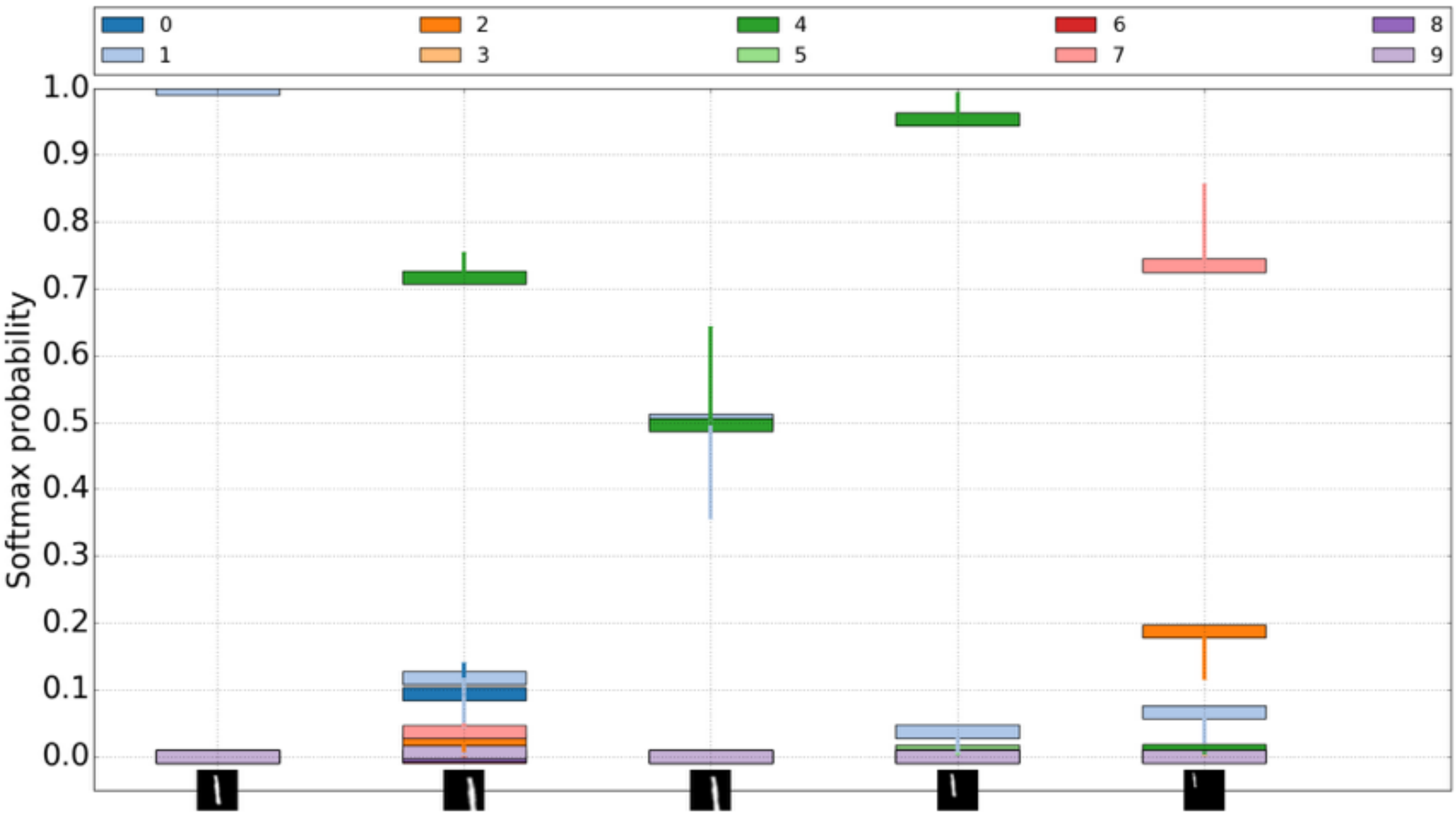


Matrix Gaussian scaling



SGLD scaling

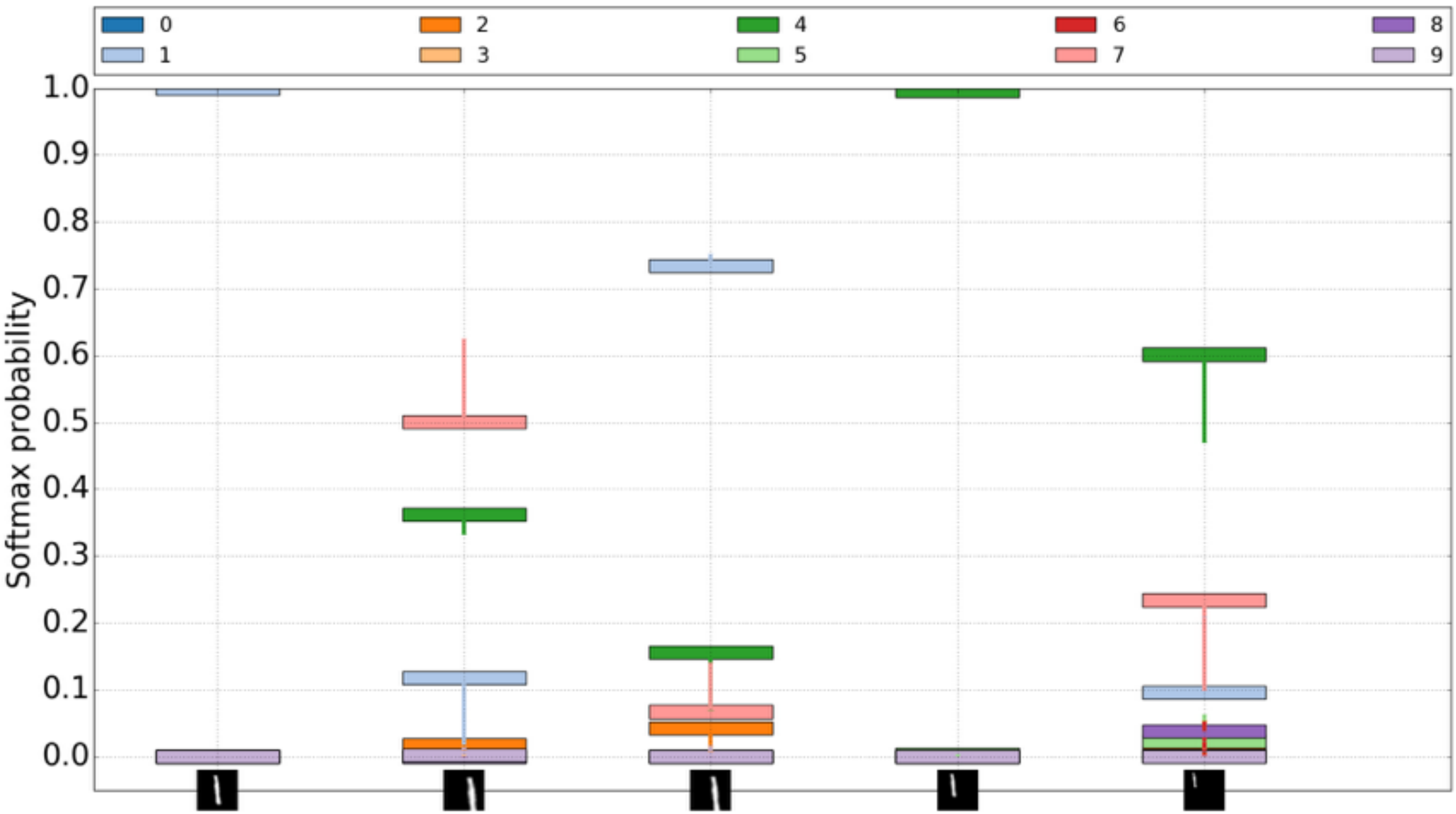
SGLD scaling



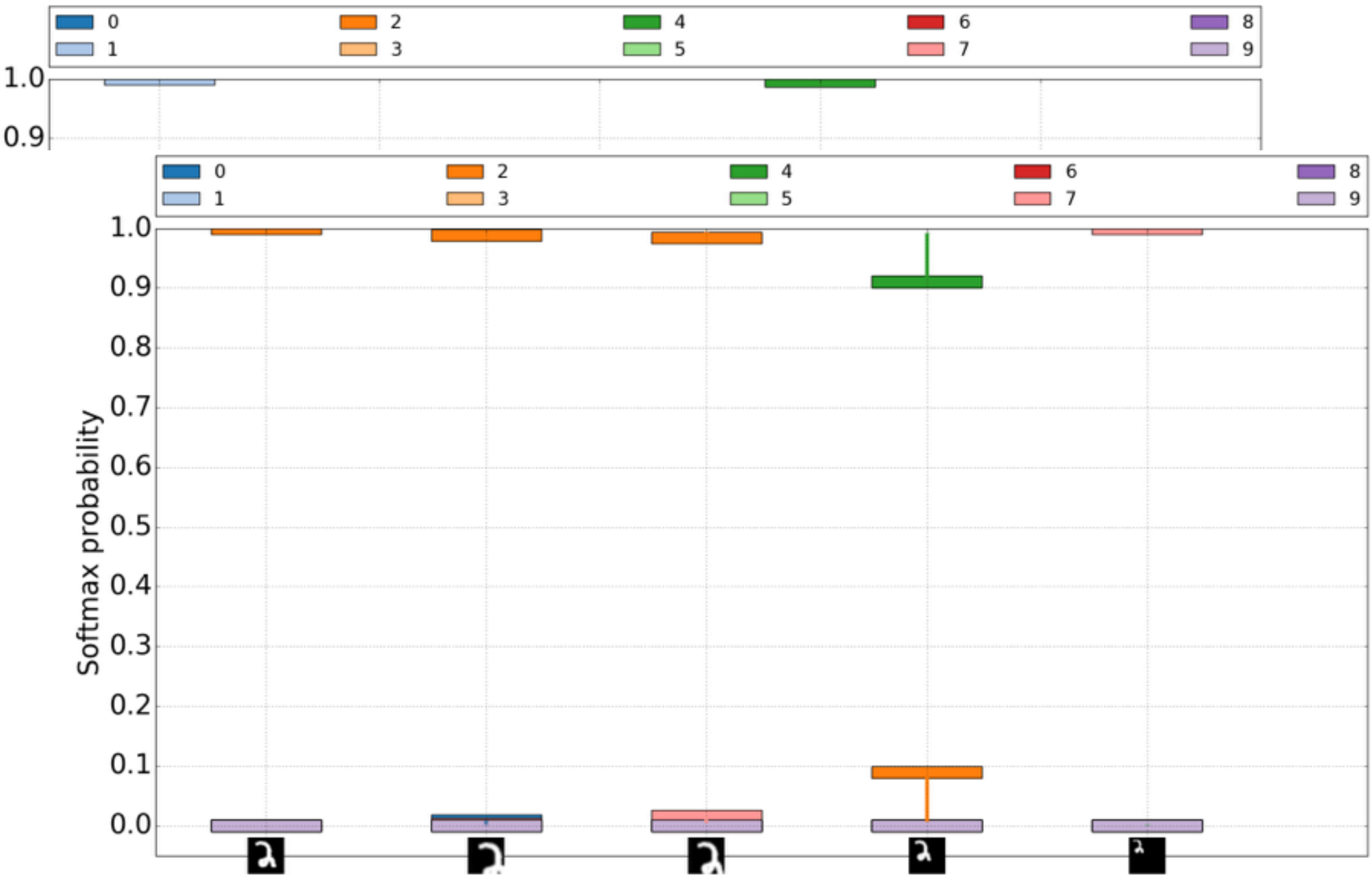
SGLD scaling

Bootstrap scaling

Bootstrap scaling



Bootstrap scaling



Verdict

Verdict

- It seems that in all circumstances the uncertainties on inputs far from the data distribution are not good

Verdict

- It seems that in all circumstances the uncertainties on inputs far from the data distribution are not good
- Deterministic weights are most affected and with Bayesian methods we do get away sometimes
  - But still in most cases we have erroneous overconfidence

Verdict

- It seems that in all circumstances the uncertainties on inputs far from the data distribution are not good
- Deterministic weights are most affected and with Bayesian methods we do get away sometimes
  - But still in most cases we have erroneous overconfidence
- In general it appears to not be a parameter prior problem as it affects even the frequentist bootstrap
  - Although a better prior might fix it

Verdict

- It seems that in all circumstances the uncertainties on inputs far from the data distribution are not good
- Deterministic weights are most affected and with Bayesian methods we do get away sometimes
    - But still in most cases we have erroneous overconfidence
- In general it appears to not be a parameter prior problem as it affects even the frequentist bootstrap
    - Although a better prior might fix it
- This suggests that the model itself (neural network) is not capable (at least as they are right now) to output reasonable probabilities

Thanks!