CONCRETE DROPOUT: EXPERIMENTS REPLICATION AND EXTENSIONS

Valerii Likhosherstov
Statistical Learning Theory Joint Master Program
Skolkovo Institute of Science and Technology
Valerii.Likhosherstov@skoltech.ru

Alfredo de la Fuente
Statistical Learning Theory Joint Master Program
Skolkovo Institute of Science and Technology
Alfredo.Delafuente@skoltech.ru

ABSTRACT

In this project we reproduce the results of the paper "Concrete Dropout" by Gal et al. ([1]). We go further by extending the range of experiments and implementing the Concrete Dropout for Natural Language Processing tasks. Furthermore, we provide an efficient implementation of a fully connected, spatial convolution and LSTM Concrete Dropout layers.

***K*eywords** bayesian deep learning, concrete dropout, model uncertainty

1 Background

Imposing a prior over the weights of a neural network and adopting a fully Bayesian inference in this rich class of estimators provides multiple advantages. First, integrating model parameters out through the MC-estimate of the integral during prediction results in a better performance [2] compared to using a point MAP estimate of weights, as it happens in vanilla-dropout [3]. Second, MC-estimates of predictions' second moments give the information about model's uncertainty on the given example [4]. Uncertainty gives a tool for human-guided model fitting, when the train set can be augmented with new examples of uncertain objects. One valuable area of application for uncertainty estimates (both epistemic and predictive) is Reinforcement Learning, where grid searching is clearly unfeasible and dropout has to be adapted dynamically, considering that the agent is expected to reduces its uncertainty as the data increases for a good model calibration. In addition, a remarkable feature of dropout's Bayesian interpretation is that with certain priors and variational approximation family, it results in sparse weight matrices [5], enabling more flexible and regularized models.

Originally, the Bayesian interpretation was only introduced for Gaussian dropout [6], where the re-parametrization trick (or pathwise derivative) allows to obtain unbiased gradient estimates with low variance. Unfortunately, pathwise derivative is not available for discrete random variables - a case of Bernoulli dropout. As an alternative solution, the log-derivative trick and REINFORCE scheme [7] suffers from high variance in stochastic gradients. For these reasons, a continuous relaxation of discrete random variables was elaborated to solve this problem - using the Gumbel-Softmax distribution [8], which converges to discrete distribution at high temperatures and is compatible with the reparametrization trick. In [8], authors exclusively employ Gumbel-Softmax to model discrete hidden states. In [1] they use a special case of Gumbel-Softmax with two possible states (Concrete distribution) to approximate Bernoulli dropout during training. Also, a correct choice of prior results in a tractable KL-divergence between prior and variational approximation, which simplifies computations.

2 Concrete Dropout

In order to overcome the computationally intensive task of grid searching over the dropout probability space, specially in scenarios of dealing with complex models with large datasets and under continuous learning settings where the epistemic uncertainty should decrease as more data is collected; the authors suggest to incorporate the dropout rate in the loss function and thus optimize directly it using gradient methods.

2.1 Derivation of the Loss function

Following the variational interpretation of dropout [9], we can see it as an approximating distribution $q_\theta(\omega)$ to the posterior in a Bayesian neural network with a set of random weight matrices $\omega = [W_l]_{l=1}^{L}$ with L layers and θ being the set of variational parameters. Then, the optimization objective can be written as follows,

$$\hat{\mathcal{L}}_{MC}(\theta) = -\frac{1}{M}\sum_{i\in S}\log p\left(y_i|f^\omega(x_i)\right) + \frac{1}{N}\mathbb{KL}\left(q_\theta(\omega)\|p(\omega)\right) \tag{1}$$

where N is the number of data points, S a random set of M data points, $f^\omega(x_i)$ the neural network's output on input x_i with parameters ω and $p(y_i|f^\omega(x_i)$ the model's likelihood (e.g. Gaussian with mean $f^\omega(x_i)$). The regularization term ensures that the approximate posterior $q_\theta(\omega)$ does not deviate too far from the prior distribution $p(\omega)$. The prior was chosen to be a discrete quantised Gaussian as described in the paper [9]. We can assume that the set of variational parameters for t he dropout

$$\mathbb{KL}\left(q_\theta(\omega)\|p(\omega)\right) = \sum_{l=1}^{L}\mathbb{KL}\left(q_{M_l}(W_l\|p(W_l)\right) \propto \frac{l^2(1-p)}{2}\|M\|^2 - K\mathcal{H}(p). \tag{2}$$

where K is the constant to balance the regularization term with the predictive term. The entropy term $\mathcal{H}(p)$ can be seen as the *dropout regularization term*, which pushes the dropout probability towards 0.5 which is the highest value it can attain; meanwhile the first term is considered as the *weight regularization term* which limits the norm of the weights used in the concrete dropout layers.

2.1.1 Concrete Distribution

One of the main restrictions, as previously mentioned, relies on the evaluation of the derivative of the objective function with respected to the dropout rate p for the loss optimization step. The preferred estimator in the literature is pathwise estimator, however we need to be able to express the random variable as a function of two parameters θ and ϵ, the latter one being a random variable which does on depend on θ. Unfortunately, the binary random variable does not admit such expression form.

In order to solve the above issue, the paper proposes a continuous relaxation of the binary discrete random variable by using the concrete distribution, expressed as follows,

$$\bar{z} = \textit{sigmoid}\left(\frac{1}{t}\left(\log\frac{p}{1-p} + \log\frac{u}{1-u}\right)\right) \tag{3}$$

where $u \sim Unif(0,1)$ and t is the temperature variable which makes the distribution closer to the discrete version.

2.2 Concrete Dropout for LSTM Layer

We give a simple extension to Concrete dropout for the case of recurrent layer - LSTM. Let's revise the definition of LSTM[1]:

$$\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})\\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})\\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})\\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})\\
c_t &= f_t c_{(t-1)} + i_t g_t\\
h_t &= o_t \tanh(c_t)
\end{aligned}$$

When using dropout with LSTM, we usually put it before the layer (at least we did that in experiments). From LSTM's definition it's clear, that this is equivalent to zeroing out columns of $W_{ii}, W_{if}W_{ig}, W_{io}$. So we only induce prior for $W_{ii}, W_{if}W_{ig}, W_{io}$ and consequently, only these matrices should be used in weight regularizer of our ELBO objective. This concludes our derivation of Concrete dropout for LSTM.

[1] https://pytorch.org/docs/stable/nn.htmltorch.nn.LSTM

3 Experiments

We analyze and validate the performance of Concrete Dropout in a wide variety of tasks.

3.1 Regression Task - UCI Datasets

In this experiment we employ Concrete dropout to solve a regression task for 3 datasets from UCI base[2]: Boston Housing Dataset, KIN8NM robotic arm dinamics dataset and Wine Quality Dataset.

We utilize a neural network of 3 fully-connected layers, each with 50 hidden units and concrete dropout. This network predicts a mean value of normally distributed output for regression. We also train a single value of log-precision parameter $\log \tau$ of output's distribution, for which we impose a prior distribution $Gamma(0.1, 0.01)$ (consequently, we add prior's logarithm to our objective function). We use weight and dropout regularizer of 10^{-2} and 10^{2} respectively.

In [1] they alternate between tuning τ and variational parameters - a so-called variational (MAP)-EM algorithm. When we replicated this setting, we have found out that dropout probabilities are hardly changing from their initial random values. Hence, we have added third pass through the data to this alternating scheme, where we only tune dropout probabilities.

In our experiments, we try to replicate a pattern of converged dropout probabilities, which is invariant to cross-validation splits. See Figure 1 for results. We observe a well-distinguished dropout probabilities between first two and the last layer (Wine dataset). For KIN8NM, first two layers converge to low dropout probabilities with small variance, while the last layer exposes a higher variability over cross-validation splits. For Boston, we do not observe any stable pattern.

3.2 MNIST

We used the standard classification dataset MNIST [10] to assess the accuracy of Concrete dropout and it's relation with the training-set size, by applying it in a Multi-Layer Perceptron (MLP) arquitecture with 3 hidden layers (with 512 units) and ReLU activations.

After trained for 100 epochs, our Concrete dropout achieves MNIST accuracy of 98% matching that of hand-tuned dropout. Figure 3 shows a decreasing converged dropout probabilities as the size of data increases for each one of the MLP layers. Furthermore, we observe in 2 that the convergence in dropout probability in first layers was achieved in significantly less number of epochs compared to the deeper ones. In addition, it validates our intuition that shallower layers in a neural network should have smaller dropout rates.

3.3 Computer Vision

FC-DenseNet model [11] shows state-of-the-art results in image segmentation task. We try to further improve the model by substituting each dropout layer in it with its concrete version.

In our experiments we use *FC-DenseNet103* topology with 103 convolution layers inside. For it we use a spatial version of Concrete dropout, when the whole channels are "zeroed-out". We opt for weight regularizer of 10^{-8} and dropout regularizer of 10^{0}. The model is trained for 100 epochs with RMSprop (learning rate - 10^{-4}). We figured out, that starting from small random dropout probabilities ($0.01 - 0.05$) results in a better performance of the model.

We compare Concrete dropout with grid-searched dropout rate of 0.2 (reported in [11]). Figure 4 reports average Intersection-over-Union score over 12 segmentation classes on training and validation set. Concrete dropout performed better on a validation set, even though training takes more time for it. If to encounter that dropout baseline required grid search, Concrete dropout appears to be more advantageous. On testing set, concrete dropout reached IoU of 0.414 and 0.418 with MC predictions against 0.401 for dropout.

Figure 6 (a) shows calibration plots for each segmentation class. Concrete dropout with MC predictions is the closest to a diagonal. Strangely, if don't use MC for predicting, it results in worse-calibrated predictions then dropout in most cases. Figure 6 (b) illustrates predictions and uncertainties.

3.4 Reinforcement Learning Task

As part of the RL setting, we tested Concrete Dropout layers under two conditions: a model-based approached which learns the dynamics of the a cartpole environment and a model-free policy gradient for learning an agent to solve the simple OpenAI 'CartPole-v0' environment. We observe how using concrete dropout improves the learning curve 7 with

[2]https://archive.ics.uci.edu/ml/datasets.html

respect to a fix dropout rate8. In general, we can show that the dropout rate decreases as more epochs we take to train the model, therefore reducing the epistemic uncertainty of predicting the next state or action for our models. For more implementation details, please refer to the github repository of the project.

3.5 Natural Language Processing

In this subsection we go beyond the scope of [1] and apply Concrete dropout to solve real-life tasks in Natural Language Processing. We comment the results from a user standpoint, i.e. like if a data scientist was trying to add Concrete dropout feature into his model.

3.5.1 Grapheme to Phoneme

We tackle the problem a grapheme-to-phoneme [3], where we have to predict English word's transcription. We applied the following architecture for our model: an encoder that consists of embedding size 100, then 3 LSTM layers with 100 hidden units and batch-norm and dropout before each. Next, we used average-pooling over encoder outputs, repeated it and put as an input to decoder - two layers of LSTM blocks same as in encoder, followed by linear layer and softmax. We trained our model for 100 epochs with RMSprop (learning rate - 10^{-3}).

For dropout, we check performance of different probabilities on a validation set. For concrete dropout, we use weight regularizer of 10^{-4} and try different values of dropout regularizer. As a metric, we use average Hamming distance between ground truth and predicted word.

Figure 10 presents learning curves for dropout and Concrete dropout. We find, that Concrete dropout performs slightly worse than dropout. We also see that Concrete dropout overfits - that happens because dropout probabilities converge to zero (Figure 9). All in all, we didn't manage to glue Concrete dropout with LSTM to obtain good results in this task.

3.5.2 Sentiment Classification Task - SST dataset

For this task we used the well-known Stanford Sentiment Treebank dataset [12][4] in its binary form. This dataset contains movie reviews which are annotated by positive or negative sentiment labels. Our proposed model is composed by one embedding layer of size 100, one LSTM layer with 256 hidden units and followed by a fully connected layer to the binary output.

We run the model with a fixed dropout rate of 0.5 to compare it against the implemented ConcreteDropout version of the LSTM layer. We used batches of 64 observations, weight regularizer 10^{-6} and dropout regularizer 2×10^{-5}.

The results after training for 20 epochs with Adam optimizer we obtained are shown at 11. We can observe that our ConcreteDropout LSTM model was able to surpass the baseline model accuracy for the validation set by taking only a few more iterations to converge. This suggests that our model is able to tune dropout rate faster, while not risking predictive accuracy.

4 Conclusions and Remarks

In this research project, we were able to validate and demonstrate how tuning the dropout rate via concrete dropout can provide us an speed up in comparison with grid-searching. One drawback of this model reflects on the need to tune hyperparameters such as weight regularizer and dropout regularizer which can dramatically affect the model predictions. Although the original paper suggests some rule of thumbs, it is still unclear how to tune the values without using a grid-search approach. We should also mention as insight, that the dropout probabilities obtained for different layers follow the general knowledge about using smaller dropout for shallow layers in a neural network. Finally, based on our reproducibility results, we can conclude that the performance of the concrete dropout layer model is comparable with the one of the model with optimal dropout probability fixed, making it much faster than grid-search with exponential number of dropout combinations.

[3] https://www.kaggle.com/c/grapheme-to-phoneme

[4] https://nlp.stanford.edu/sentiment/treebank.html

References

[1] Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3581–3590. Curran Associates, Inc., 2017.

[2] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *CoRR*, abs/1506.02158, 2015.

[3] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[4] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1050–1059. JMLR.org, 2016.

[5] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2498–2507, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[6] Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2575–2583. Curran Associates, Inc., 2015.

[7] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992.

[8] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. 2017.

[9] Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

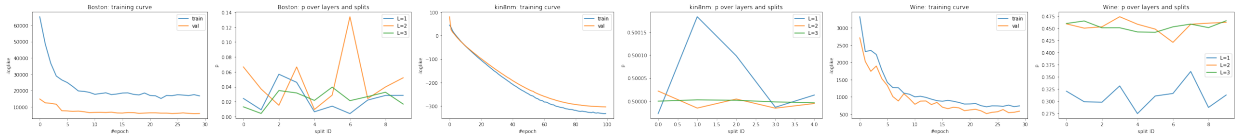
[10] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[11] Simon Jegou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. *arXiv e-prints*, abs/1611.09326, 2016.

[12] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

5 Appendix

In this section we display all relevant plots of results referenced in the experimental section.



(a) Boston (b) KIN8NM (c) Wine

Figure 1: UCI experiments. Best viewed on a computer screen.

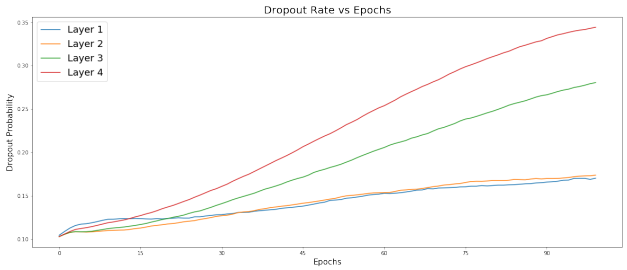


Figure 2: Convergence of Dropout Probabilities per Layer with respect to training epochs.

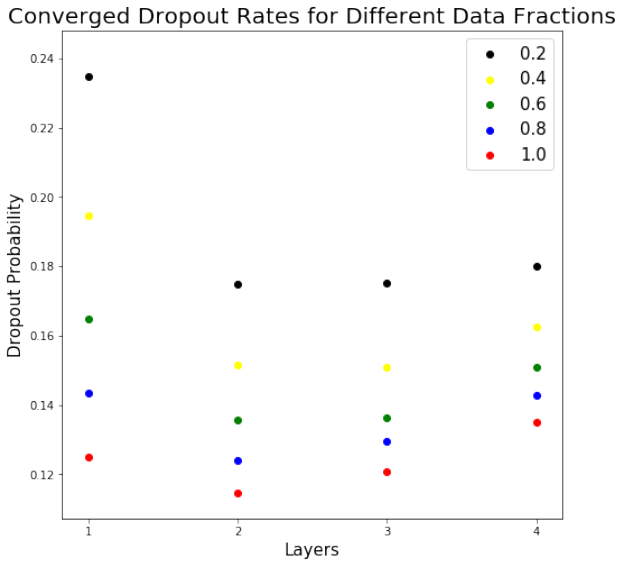


Figure 3: Convergence of Dropout Probabilities per Layer given different fractions of training set.

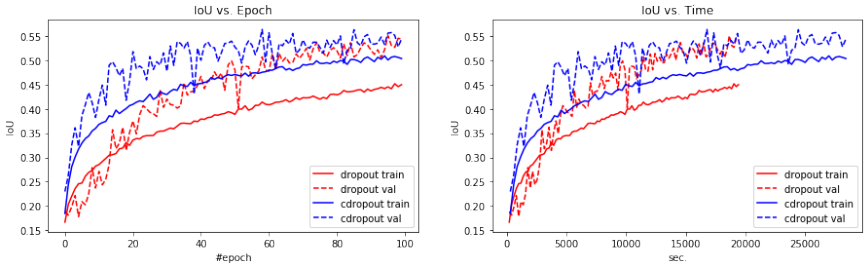


Figure 4: Segmentation: learning curves.

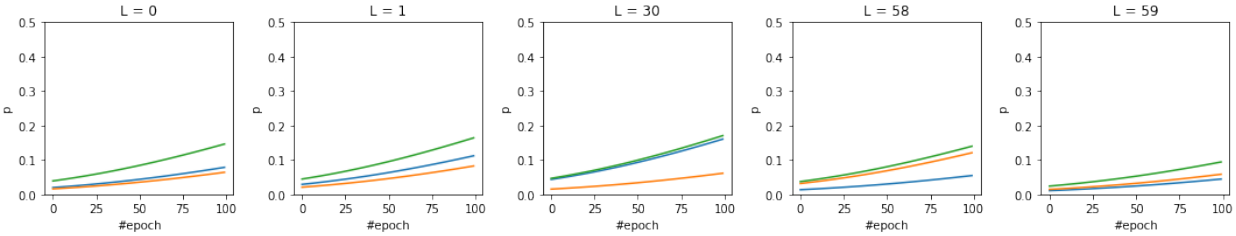


Figure 5: Segmentation: dropout rate convergence over layers and several data fits.

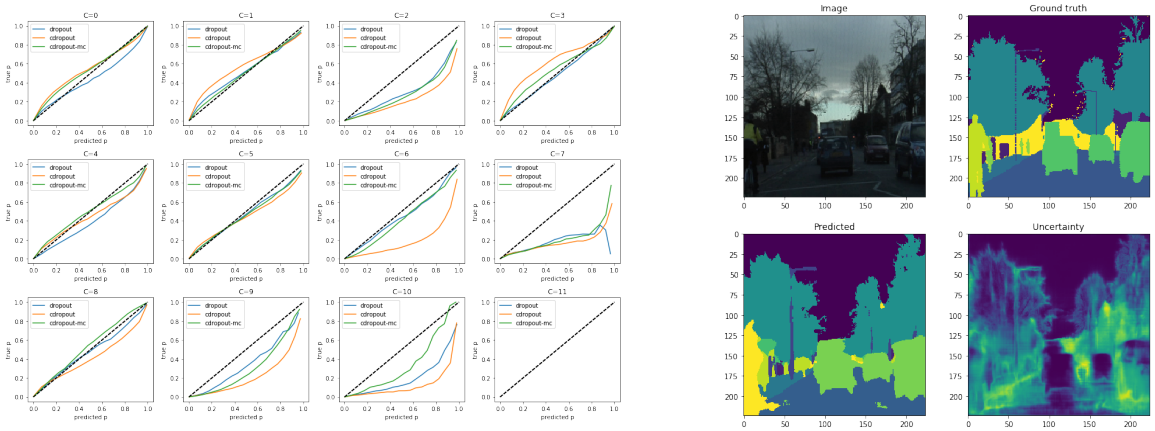


Figure 6: Calibration plots for 12 classes (a) and example of segmentation and uncertainty (b).

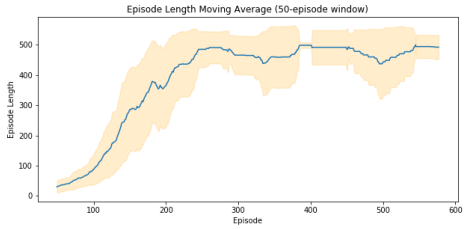


Figure 7: Learning Curve with Concrete Dropout for CartPole environment.

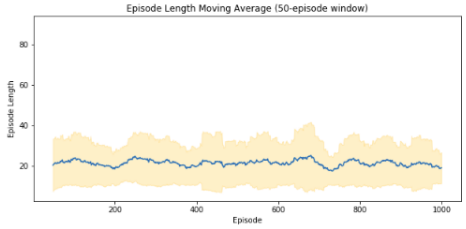


Figure 8: Learning Curve with Fixed dropout (0.5) for CartPole environment.

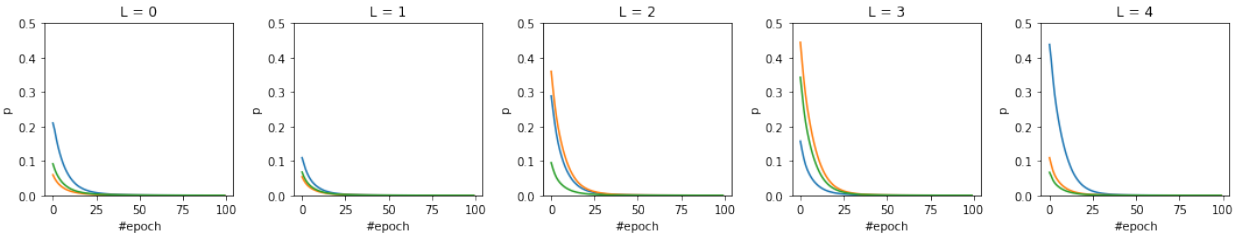


Figure 9: Grapheme-to-Phoneme: dropout rate convergence.

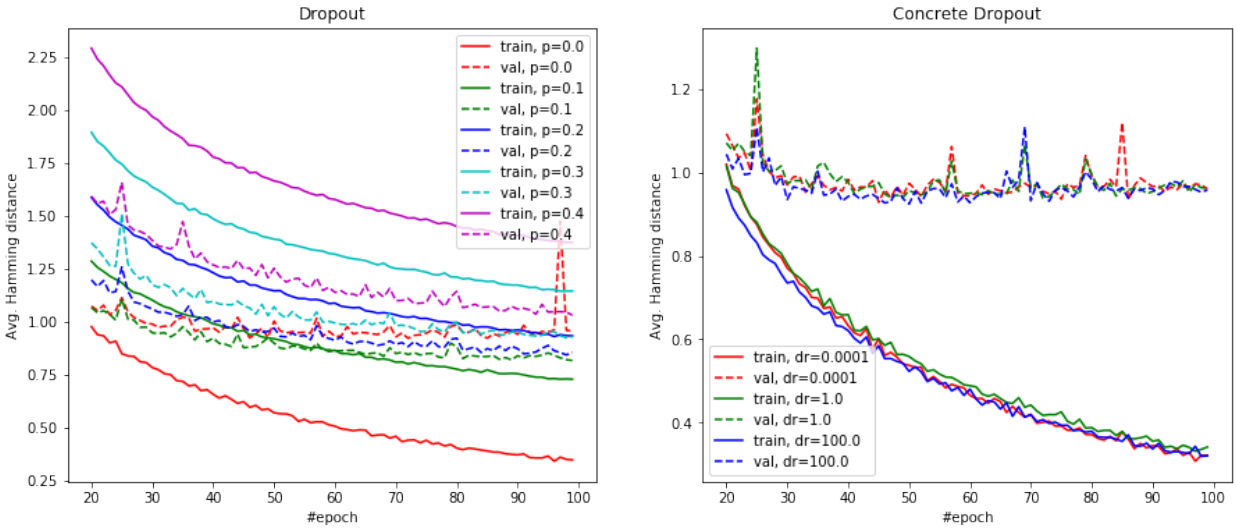
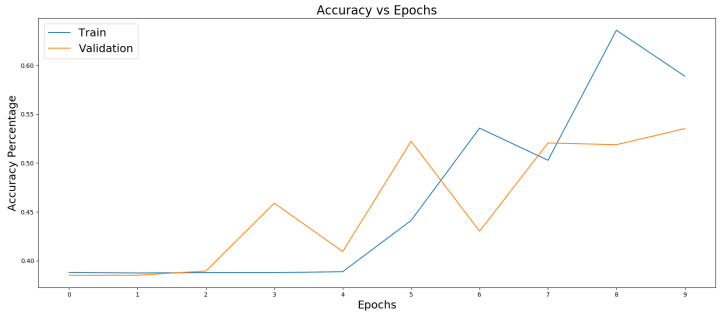
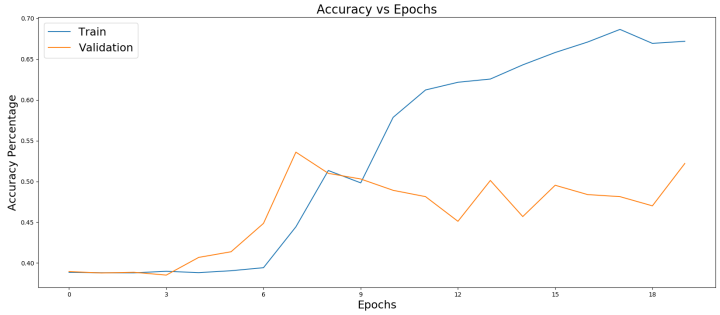


Figure 10: Grapheme-to-Phoneme: learning curves.



(a)



(b)

Figure 11: (a) Sentiment Classification learning curve for constant dropout (0.5). (b) Sentiment Classification learning curve for concrete dropout.