

Contextual Dropout for Bayesian Neural Networks: Appendix

A DETAILS OF ARM GRADIENT ESTIMATOR FOR BERNOULLI CONTEXTUAL DROPOUT

In this section, we will explain the implementation details of ARM for Bernoulli contextual dropout. To compute the gradients with respect to the parameters of the variational distribution, a commonly used gradient estimator is the REINFORCE estimator (Williams, 1992) as

$$\nabla_{\varphi} \mathcal{L}(\mathbf{x}, y) = \mathbb{E}_{\mathbf{z} \sim q_{\phi}(\cdot | \mathbf{x})} [r(\mathbf{x}, \mathbf{z}, y) \nabla_{\varphi} \log q_{\phi}(\mathbf{z} | \mathbf{x})], \quad r(\mathbf{x}, \mathbf{z}, y) := \log \frac{p_{\theta}(y | \mathbf{x}, \mathbf{z}) p_{\eta}(\mathbf{z})}{q_{\phi}(\mathbf{z} | \mathbf{x})}.$$

This gradient estimator is, however, known to have high variance (Yin & Zhou, 2018). To mitigate this issue, we use ARM to compute the gradient with Bernoulli random variable.

ARM gradient estimator: In general, denoting $\sigma(\alpha) = 1/(1 + e^{-\alpha})$ as the sigmoid function, ARM expresses the gradient of $\mathcal{E}(\alpha) = \mathbb{E}_{\mathbf{z} \sim \prod_{k=1}^K \text{Ber}(z_k; \sigma(\alpha_k))} [r(\mathbf{z})]$ as

$$\nabla_{\alpha} \mathcal{E}(\alpha) = \mathbb{E}_{\boldsymbol{\pi} \sim \prod_{k=1}^K \text{Uniform}(\pi_k; 0, 1)} [g_{\text{ARM}}(\boldsymbol{\pi})], \quad g_{\text{ARM}}(\boldsymbol{\pi}) := [r(\mathbf{z}_{\text{true}}) - r(\mathbf{z}_{\text{sudo}})](1/2 - \boldsymbol{\pi}), \quad (6)$$

where $\mathbf{z}_{\text{true}} := \mathbf{1}_{[\boldsymbol{\pi} < \boldsymbol{\sigma}(\alpha)]}$ and $\mathbf{z}_{\text{sudo}} := \mathbf{1}_{[\boldsymbol{\pi} > \boldsymbol{\sigma}(-\alpha)]}$ are referred to as the true and pseudo actions, respectively, and $\mathbf{1}_{[\cdot]} \in \{0, 1\}^K$ is an indicator function.

Sequential ARM: Note that the above equation is not directly applicable to our model due to the cross-layer dependence. However, the dropout masks within each layer are independent of each other conditioned on these of the previous layers, so we can break our expectation into a sequence and apply ARM sequentially. We rewrite $\mathcal{L} = \mathbb{E}_{\mathbf{z} \sim q_{\phi}(\cdot | \mathbf{x})} [r(\mathbf{x}, \mathbf{z}, y)]$. When computing $\nabla_{\varphi} \mathcal{L}$, we can ignore the φ in r as the expectation of $\nabla_{\varphi} \log q_{\phi}(\mathbf{z} | \mathbf{x})$ is zero. Using the chain rule, we have $\nabla_{\varphi} \mathcal{L} = \sum_{l=1}^L \nabla_{\alpha^l} \mathcal{L} \nabla_{\varphi} \alpha^l$. With decomposition $\mathcal{L} = \mathbb{E}_{\mathbf{z}^{1:l-1} \sim q_{\phi}(\cdot | \mathbf{x})} \mathbb{E}_{\mathbf{z}^l \sim \text{Ber}(\sigma(\alpha^l))} [r(\mathbf{x}, \mathbf{z}^{1:l}, y)]$, where $r(\mathbf{x}, \mathbf{z}^{1:l}, y) := \mathbb{E}_{\mathbf{z}^{l+1:L} \sim q_{\phi}(\cdot | \mathbf{x}, \mathbf{z}^{1:l})} [r(\mathbf{x}, \mathbf{z}, y)]$, we know

$$\begin{aligned} \nabla_{\alpha^l} \mathcal{L} &= \mathbb{E}_{\mathbf{z}^{1:l-1} \sim q_{\phi}(\cdot | \mathbf{x})} \mathbb{E}_{\boldsymbol{\pi}^l \sim \prod_k \text{Uniform}(\pi_k^l; 0, 1)} [g_{\text{ARM}}(\boldsymbol{\pi}^l)], \\ g_{\text{ARM}}(\boldsymbol{\pi}^l) &= [r(\mathbf{x}, \mathbf{z}^{1:l-1}, \mathbf{z}_{\text{true}}^l, y) - r(\mathbf{x}, \mathbf{z}^{1:l-1}, \mathbf{z}_{\text{sudo}}^l, y)](1/2 - \boldsymbol{\pi}^l), \end{aligned}$$

where $\mathbf{z}_{\text{true}}^l := \mathbf{1}_{[\boldsymbol{\pi}^l < \boldsymbol{\sigma}(\alpha^l)]}$ and $\mathbf{z}_{\text{sudo}}^l := \mathbf{1}_{[\boldsymbol{\pi}^l > \boldsymbol{\sigma}(-\alpha^l)]}$. We estimate the gradients via Monte Carlo integration. We provide the pseudo code in Algorithm 1.

Implementation details: The computational complexity of sequential ARM is $O(L)$ times of that of the decoder computation. Although it is embarrassingly parallelizable, in practice, with limited computational resource available, it may be challenging to use sequential ARM when L is fairly large. In such cases, the original non-sequential ARM can be viewed as an approximation to strike a good balance between efficiency and accuracy (see the pseudo code in Algorithm 2 in Appendix). In our cases, for image classification models, L is small enough (3 for MLP, 12 for WRN) for us to use sequential ARM. For VQA, L is as large as 62 and hence we choose the non-sequential ARM.

To control the learning rate of the encoder, we use a scaled sigmoid function: $\sigma_t(\alpha^l) = \frac{1}{1 + \exp(-t\alpha^l)}$, where a larger t corresponding to a larger learning rate for the encoder. This function is also used in Li & Ji (2019) to facilitate the transition of probability between 0 and 1 for the purpose of pruning NN weights.

B ALGORITHMS

Below, we present training algorithms for both Bernoulli and Gaussian contextual dropout.

Algorithm 1: Bernoulli contextual dropout with sequential ARM

Input: data \mathcal{D} , r , $\{g_{\theta}^l\}_{l=1}^L$, $\{h_{\varphi}^l\}_{l=1}^L$, step size s
Output: updated θ , φ , η

repeat
 $G_{\varphi} = 0$;
 Sample x, y from data \mathcal{D} ;
 $x^0 = x$
for $l = 1$ **to** L **do**
 $U^l = g_{\theta}^l(x^{l-1})$, $\alpha^l = h_{\varphi}^l(U^l)$
 Sample π^l from Uniform(0,1);
 $z_{\text{true}}^l := \mathbf{1}_{[\pi^l < \sigma_t(\alpha^l)]}$;
 $z_{\text{sudo}}^l := \mathbf{1}_{[\pi^l > \sigma_t(-\alpha^l)]}$;
if $z_{\text{true}}^l = z_{\text{sudo}}^l$ **then**
 $r_{\text{sudo}}^l = \text{None}$;
else
 $x_{\text{sudo}}^l = U^l \odot z_{l,\text{sudo}}$
for $k = l + 1$ **to** L **do**
 $U_{\text{sudo}}^k = g_{\theta}^k(x_{\text{sudo}}^{k-1})$, $\alpha_{\text{sudo}}^k = h_{\varphi}^k(U_{\text{sudo}}^k)$
 Sample π_{sudo}^k from Uniform(0,1);
 $z_{\text{sudo}}^k := \mathbf{1}_{[\pi_{\text{sudo}}^k < \sigma_t(\alpha_{\text{sudo}}^k)]}$;
 $x_{\text{sudo}}^k = U_{\text{sudo}}^k \odot z_{k,\text{sudo}}$;
end for
 $r_{\text{sudo}}^l = r(x_{\text{sudo}}^L, y)$
end if
 $x^l = U^l \odot z_{\text{true}}^l$
end for
 $r_{\text{true}} = r(x_{\text{true}}^L, y)$
for $l = 1$ **to** L **do**
if r_{sudo}^l is not None **then**
 $G_{\varphi} = G_{\varphi} + t(r_{\text{true}} - r_{\text{sudo}}^l)(1/2 - \pi^l)\nabla_{\varphi}\alpha^l$;
end if
end for
 $\varphi = \varphi + sG_{\varphi}$, with step-size s ;
 $\theta = \theta + s \frac{\partial \log p_{\theta}(y | x, z_{1:L,\text{true}})}{\partial \theta}$;
 $\eta = \eta + s \frac{\partial \log p_{\eta}(z_{1:L,\text{true}})}{\partial \eta}$;
until convergence

Algorithm 2: Bernoulli contextual dropout with independent ARM

Input: data \mathcal{D} , r , $\{g_{\theta}^l\}_{l=1}^L$, $\{h_{\varphi}^l\}_{l=1}^L$, step size s
Output: updated θ , φ , η

repeat

$G_{\varphi} = 0$;
 Sample x, y from data \mathcal{D} ;
 $x^0 = x$

for $l = 1$ **to** L **do**

$U^l = g_{\theta}^l(x^{l-1})$, $\alpha^l = h_{\varphi}^l(U^l)$
 Sample π^l from Uniform(0,1);
 $z_{\text{true}}^l := \mathbf{1}_{[\pi^l < \sigma_t(\alpha^l)]}$;
 $x^l = U^l \odot z_{\text{true}}^l$

end for

$r_{\text{true}} = r(x_{\text{true}}^L, y)$
 $x_{\text{sudo}}^0 = x$

for $l = 1$ **to** L **do**

$U_{\text{sudo}}^l = g_{\theta}^l(x_{\text{sudo}}^{l-1})$, $\alpha_{\text{sudo}}^l = h_{\varphi}^l(U_{\text{sudo}}^l)$
 $z_{\text{sudo}}^l := \mathbf{1}_{[\pi_{\text{sudo}}^l > \sigma_t(-\alpha_{\text{sudo}}^l)]}$;
 $x_{\text{sudo}}^l = U_{\text{sudo}}^l \odot z_{\text{sudo}}^l$

end for

$r_{\text{sudo}} = r(x_{\text{sudo}}^L, y)$;

for $l = 1$ **to** L **do**

$G_{\varphi} = G_{\varphi} + t(r_{\text{true}} - r_{\text{sudo}})(1/2 - \pi^l)\nabla_{\varphi}\alpha^l$;

end for

$\varphi = \varphi + sG_{\varphi}$, with step-size s ;

$\theta = \theta + s \frac{\partial \log p_{\theta}(y | x, z_{1:L, \text{true}})}{\partial \theta}$;

$\eta = \eta + s \frac{\partial \log p_{\eta}(z_{1:L, \text{true}})}{\partial \eta}$;

until convergence

Algorithm 3: Gaussian contextual dropout with reparamaterization trick

Input: data \mathcal{D} , r , $\{g_{\theta}^l\}_{l=1}^L$, $\{h_{\varphi}^l\}_{l=1}^L$, step size s
Output: updated θ , φ , η

repeat
 Sample x, y from data \mathcal{D} ;
 $x^0 = x$
for $l = 1$ **to** L **do**
 $U^l = g_{\theta}^l(x^{l-1})$, $\alpha^l = h_{\varphi}^l(U^l)$
 Sample ϵ^l from $\mathcal{N}(0, 1)$;
 $\tau^l = \sqrt{\frac{1 - \sigma_t(\alpha^l)}{\sigma_t(\alpha^l)}}$;
 $z^l := \mathbf{1} + \tau^l \odot \epsilon^l$;
 $x^l = U^l \odot z^l$
end for
 $\varphi = \varphi + s \nabla_{\varphi} (\log p_{\theta}(y | \mathbf{x}, \mathbf{z}_{1:L}) - \frac{\log q_{\phi}(\mathbf{z}_{1:L} | \mathbf{x})}{\log p_{\eta}(\mathbf{z}_{1:L})})$, with step-size s ;
 $\theta = \theta + s \frac{\partial \log p_{\theta}(y | \mathbf{x}, \mathbf{z}_{1:L})}{\partial \theta}$;
 $\eta = \eta + s \frac{\partial \log p_{\eta}(\mathbf{z}_{1:L})}{\partial \eta}$;
until convergence

C DETAILS OF EXPERIMENTS

All experiments are conducted using a single Nvidia Tesla V100 GPU.

Table 5: Model size comparison among different methods.

METHOD	MLP	WRN	MCAN	RESNET-18
MC OR CONCRETE	267K	36.5M	58M	11.6M
CONTEXTUAL	311K	36.6M	61M	11.8M
BAYES BY BACKPROP	534K	-	-	-

C.1 IMAGE CLASSIFICATION

MLP: We consider an MLP with two hidden layers of size 300 and 100, respectively, and use ReLU activations. Dropout is applied to all three full-connected layers. We use MNIST as the benchmark. All models are trained for 200 epochs with batch size 128 and the Adam optimizer (Kingma & Ba, 2014) ($\beta_1 = 0.9$, $\beta_2 = 0.999$). The learning rate is 0.001. We compare contextual dropout with MC dropout (Gal & Ghahramani, 2016) and concrete dropout (Gal et al., 2017). For MC dropout, we use the hand-tuned dropout rate at 0.2. For concrete dropout, we initialize the dropout rate at 0.2 for Bernoulli dropout and the standard deviation parameter at 0.5 for Gaussian dropout. and set the Concrete temperature at 0.1 (Gal et al., 2017). We initialize the weights in contextual dropout with *He-initialization* preserving the magnitude of the variance of the weights in the forward pass (He et al., 2015). We initialize the biases in the way that the dropout rate is 0.2 when the weights for contextual dropout are zeros. We also initialize our prior dropout rate at 0.2. For hyperparameter tuning, we hold out 10,000 samples randomly selected from the training set for validation. We use the chosen hyperparameters to train on the full training set (60,000 samples) and evaluate on the testing set (10,000 samples). We use Leaky ReLU (Xu et al., 2015a) with 0.1 as the non-linear operator in contextual dropout. The reduction ratio γ is set as 10, and sigmoid scaling factor t as 0.01. For Bayes by Backprop, we use $-\log \sigma_1 = 0$, $-\log \sigma_2 = 6$, $\pi = 0.2$ (following the notation in the original paper). For evaluation, we set $M = 20$.

WRN: We consider WRN (Zagoruyko & Komodakis, 2016), including 25 convolutional layers. In Figure 6, we show the architecture of WRN, where dropout is applied to the first convolutional layer in each network block; in total, dropout is applied to 12 convolutional layers. We use CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009) as benchmarks. All experiments are trained for 200 epochs with the Nesterov Momentum optimizer (Nesterov, 1983), whose base learning rate is set as 0.1, with

decay factor $1/5$ at epochs 60 and 120. All other hyperparameters are the same as MLP except for Gaussian dropout, where we use standard deviation equal to 0.8 for the CIFAR100 with no noise and 1 for all other cases.

ResNet: We used ResNet-18 as the baseline model. We use momentum SGD, with learning rate 0.1, and momentum weight 0.9. Weight decay is utilized with weight $1e^{-4}$. For models trained from scratch, we train the models with 90 epochs. For finetuning models, we start with pretrained baseline ResNet models and finetune for 1 epoch.

Group Name	Layers
conv1	[Original Conv (16)]
conv2	[Conv + Dropout (160); Original Conv (160)] x 4
conv3	[Conv + Dropout (320); Original Conv (320)] x 4
conv4	[Conv + Dropout (640); Original Conv (640)] x 4

Figure 6: Architecture of the Wide Residual Network.

C.2 VQA

Dataset: The dataset is split into the training (80k images and 444k QA pairs), validation (40k images and 214k QA pairs), and testing (80k images and 448k QA pairs) sets. We perform evaluation on the validation set as the true labels for the test set are not publicly available (Deng et al., 2018).

Evaluation metric: the evaluation for VQA is different from image classification. The accuracy for a single answer could be a number between 0 and 1 (Goyal et al., 2017): $\text{Acc}(ans) = \min\{(\#\text{human that said } ans)/3, 1\}$. We generalize the uncertainty evaluation accordingly:

$$n_{ac} = \sum_i \text{Acc}_i \text{Cer}_i, n_{iu} = \sum_i (1 - \text{Acc}_i)(1 - \text{Cer}_i), n_{au} = \sum_i \text{Acc}_i(1 - \text{Cer}_i), n_{ic} = \sum_i (1 - \text{Acc}_i)(\text{Cer}_i)$$

where for the i th prediction Acc_i is the accuracy and $\text{Cer}_i \in \{0, 1\}$ is the certainty indicator.

Experimental setting: We follow the setting by Yu et al. (2019), where bottom-up features extracted from images by Faster R-CNN (Ren et al., 2015) are used as visual features, pretrained word-embeddings (Pennington et al., 2014) and LSTM (Hochreiter & Schmidhuber, 1997) are used to extract question features. We adopt the encoder-decoder structure in MCAN with six co-attention layers. We use the same model hyperparameters and training settings in Yu et al. (2019) as follows: the dimensionality of input image features, input question features, and fused multi-modal features are set to be 2048, 512, and 1024, respectively. The latent dimensionality in the multi-head attention is 512, the number of heads is set to 8, and the latent dimensionality for each head is 64. The size of the answer vocabulary is set to $N = 3129$ using the strategy in Teney et al. (2018). To train the MCAN model, we use the Adam optimizer (Kingma & Ba, 2014) with $\beta_1 = 0.9$ and $\beta_2 = 0.98$. The base learning rate is set to $\min(2.5te^{-5}, 1e^{-4})$, where t is the current epoch number starting from 1. After 10 epochs, the learning rate is decayed by $1/5$ every 2 epochs. All the models are trained up to 13 epochs with the same batch size of 64.

We only conduct training on the training set (no data augmentation with visual genome dataset), and evaluation on the validation set. For MC dropout, we use the dropout rate of 0.1 for Bernoulli dropout as in Yu et al. (2019) and the standard deviation parameter of $1/3$ for Gaussian dropout. For concrete dropout, we initialize the dropout rate at 0.1 and set the Concrete temperature at 0.1 (Gal et al., 2017). For hyperparameter tuning, we randomly hold out 20% of the training set for validation. After tuning, we train on the whole training set and evaluate on the validation set. We initialize the weights with *He-initialization* preserving the magnitude of the variance of the weights in the forward pass (He et al., 2015). We initialize the biases in the way that the dropout rate is 0.1 when the weights for contextual dropout are zeros. We also initialize our prior dropout rate at 0.1. We use ReLU as the non-linear operator in contextual dropout. We use $\gamma = 32$ for layers with $C_d^l > 32$, $\gamma = 8$ for layers with $8 < C_d^l \leq 32$, otherwise $\gamma = 1$. We set $\alpha \in \mathbb{R}^{d_v}$ for residual layers.

D STATISTICAL TEST FOR UNCERTAINTY ESTIMATION

Consider M posterior samples of predictive probabilities $\{\mathbf{p}_m\}_{m=1}^M$, where \mathbf{p}_m is a vector with the same dimension as the number of classes. For single-label classification models, \mathbf{p}_m is produced

by a softmax layer and sums to one, while for multi-label classification models, p_m is produced by a sigmoid layer and each element is between 0 and 1. The former output is used in most image classification models, while the latter is often used in VQA where multiple answers could be true for a single input. In both cases, to quantify how confident our model is about this prediction, we evaluate whether the difference between the probabilities of the first and second highest classes is statistically significant with a statistical test. We conduct the normality test on the output probabilities for both image classification and VQA models, and find most of the output probabilities are approximately normal (we randomly pick some Q-Q plots (Ghasemi & Zahediasl, 2012) and show them in Figures 7 and 8). This motivates us to use two-sample t -test². In the following, we briefly summarize the two-sample t -test we use.

Two sample hypothesis testing is an inferential statistical test that determines whether there is a statistically significant difference between the means in two groups. The null hypothesis for the t -test is that the population means from the two groups are equal: $\mu_1 = \mu_2$, and the alternative hypothesis is $\mu_1 \neq \mu_2$. Depending on whether each sample in one group can be paired with another sample in the other group, we have either paired t -test or independent t -test. In our experiments, we utilize both types of two sample t -test. For a single-label model, the probabilities are dependent between two classes due to the softmax layer, therefore, we use the *paired* two-sample t -test; for a multi-label model, the probabilities are independent given the logits of the output layer, so we use the *independent* two-sample t -test.

For paired two-sample t -test, we calculate the difference between the paired observations calculate the t -statistic as below:

$$T = \frac{\bar{Y}}{s/\sqrt{N}},$$

where \bar{Y} is the mean difference between the paired observations, s is the standard deviation of the differences, and N is the number of observations. Under the null hypothesis, this statistic follows a t -distribution with $N - 1$ degrees of freedom if the difference is normally distributed. Then, we use this t -statistic and t -distribution to calculate the corresponding p -value.

For independent two-sample t -test, we calculate the t -statistic as below:

$$T = \frac{\bar{Y}_1 - \bar{Y}_2}{\sqrt{s^2/N_1 + s^2/N_2}}$$

$$s^2 = \frac{\sum(y_1 - \bar{Y}_1) + \sum(y_2 - \bar{Y}_2)}{N_1 + N_2 - 2}$$

where N_1 and N_2 are the sample sizes, and \bar{Y}_1 and \bar{Y}_2 are the sample means. Under the null hypothesis, this statistic follows a t -distribution with $N_1 + N_2 - 2$ degrees of freedom if both y_1 and y_2 are normally distributed. We calculate the p -value accordingly.

To justify the assumption of the two-sample t -test, we run the normality test on the output probabilities for both image classification and VQA models. We find most of the output probabilities are approximately normal. We randomly pick some Q-Q plots (Ghasemi & Zahediasl, 2012) and show them in Figures 7 and 8.

E TABLES AND FIGURES FOR p -VALUE 0.01, 0.05 AND 0.1

Table 6: Complete results on MNIST with MLP

	ORIGINAL DATA		NOISY DATA	
	ACCURACY	PAVPU(0.01 / 0.05 / 0.1)	ACCURACY	PAVPU(0.01 / 0.05 / 0.1)
MC DROPOUT - BERNOULLI	98.62	98.25 / 98.39 / 98.44	86.36	84.29 / 85.63 / 86.10
MC DROPOUT - GAUSSIAN	98.67	98.23 / 98.41 / 98.46	86.31	83.99 / 85.64 / 86.03
CONCRETE DROPOUT	98.61	98.43 / 98.50 / 98.57	86.52	85.98 / 86.77 / 86.92
BAYES BY BACKPROP	98.44	98.26 / 98.42 / 98.56	86.55	86.89 / 87.13 / 87.26
BERNOULLI CONTEXTUAL DROPOUT	99.08 (0.04)	98.74 (0.17) / 98.92 (0.08) / 99.09 (0.08)	87.43 (0.39)	87.75 (0.24) / 87.81 (0.23) / 87.89 (0.25)
GAUSSIAN CONTEXTUAL DROPOUT	98.92(0.09)	98.71(0.02) / 98.90(0.08) / 99.03(0.07)	87.35(0.33)	87.64(0.19) / 87.72(0.29) / 87.78(0.32)

²Note that we also tried a nonparametric test, Wilcoxon rank-sum test, and obtain similar results.

Table 7: Loglikelihood on original MNIST with MLP.

	LOG LIKELIHOOD
MC - BERNOULLI	-1.4840 \pm 0.0004
MC - GAUSSIAN	-1.4820 \pm 0.0003
CONCRETE	-1.4822 \pm 0.0012
BAYES BY BACKPROP	-1.4806 \pm 0.0007
BERNOULLI CONTEXTUAL	-1.4537 \pm 0.0005
GAUSSIAN CONTEXTUAL	-1.4589 \pm 0.0005

Table 8: Complete results on CIFAR-10 with WRN

	ORIGINAL DATA		NOISY DATA	
	ACCURACY	PAVPU(0.01 / 0.05 / 0.1)	ACCURACY	PAVPU(0.01 / 0.05 / 0.1)
MC DROPOUT - BERNOULLI	94.58	78.73 / 82.34 / 84.21	79.51	72.89 / 74.43 / 75.04
MC DROPOUT - GAUSSIAN	93.81	92.59 / 93.24 / 93.85	79.33	80.43 / 81.24 / 82.31
CONCRETE DROPOUT	94.60	73.51 / 78.41 / 81.01	79.34	72.72 / 73.89 / 74.72
BERNOULLI CONTEXTUAL DROPOUT	95.92(0.10)	95.25(0.23) / 95.74(0.12) / 96.02(0.16)	81.49(0.19)	82.56 (0.50) / 83.28 (0.31) / 83.91 (0.28)
GAUSSIAN CONTEXTUAL DROPOUT	96.04 (0.1)	95.42 (0.07) / 95.85 (0.07) / 96.10 (0.06)	81.64 (0.31)	82.38 (0.41) / 82.80(0.36) / 83.43(0.36)

Table 9: Complete log likelihood results on CIFAR-10 with WRN

	CIFAR-10	
	ORIGINAL DATA	NOISY DATA
MC DROPOUT - BERNOULLI	-1.91	-1.93
MC DROPOUT - GAUSSIAN	-1.54	-1.72
CONCRETE DROPOUT	-1.98	-2.0
BERNOULLI CONTEXTUAL DROPOUT	-1.24	-1.47
GAUSSIAN CONTEXTUAL DROPOUT	-1.19	-1.51

Table 10: Complete results on CIFAR-100 with WRN

	ORIGINAL DATA		NOISY DATA	
	ACCURACY	PAVPU(0.01 / 0.05 / 0.1)	ACCURACY	PAVPU(0.01 / 0.05 / 0.1)
MC DROPOUT - BERNOULLI	79.03	56.90 / 61.54 / 64.14	52.01	53.86 / 54.25 / 54.63
MC DROPOUT - GAUSSIAN	76.63	77.35 / 78.05 / 78.26	51.38	56.83 / 57.02 / 57.31
CONCRETE DROPOUT	79.19	59.45 / 64.14 / 66.63	51.38	57.62 / 56.61 / 55.89
BERNOULLI CONTEXTUAL DROPOUT	80.85(0.05)	81.04(0.28) / 81.56(0.31) / 81.86(0.21)	53.64(0.45)	58.29 (0.30) / 58.63 (0.50) / 59.36 (0.49)
GAUSSIAN CONTEXTUAL DROPOUT	80.93 (0.18)	81.43 (0.1) / 81.69 (0.16) / 82.02 (0.14)	53.72 (0.34)	58.01(0.6) / 58.49(0.43) / 58.95(0.37)

F QUALITATIVE ANALYSIS

In this section, we include the Q-Q plots of the output probabilities as the normality test for the assumptions of two-sample t -test. In Figure 7, we test the normality of differences between highest probabilities and second highest probabilities on WRN model with contextual dropout trained on the original CIFAR-10 dataset. In Figure 8, we test the normality of highest probabilities and second highest probabilities (separately) on VQA model with contextual dropout trained on the original VQA-v2 dataset. We use 20 data points for the plots.

F.1 NORMALITY TEST OF OUTPUT PROBABILITIES

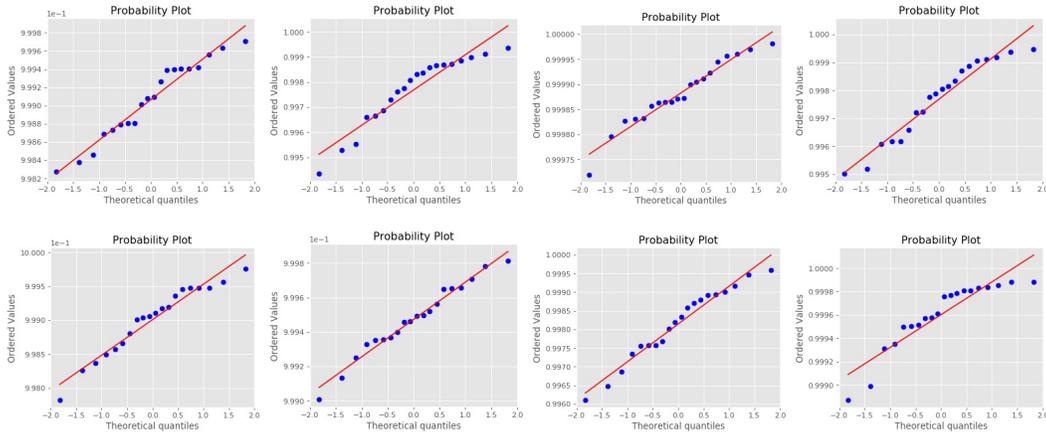


Figure 7: QQ Plot for differences between highest probabilities and second highest probabilities on WRN model with contextual dropout trained on the original CIFAR-10 dataset.

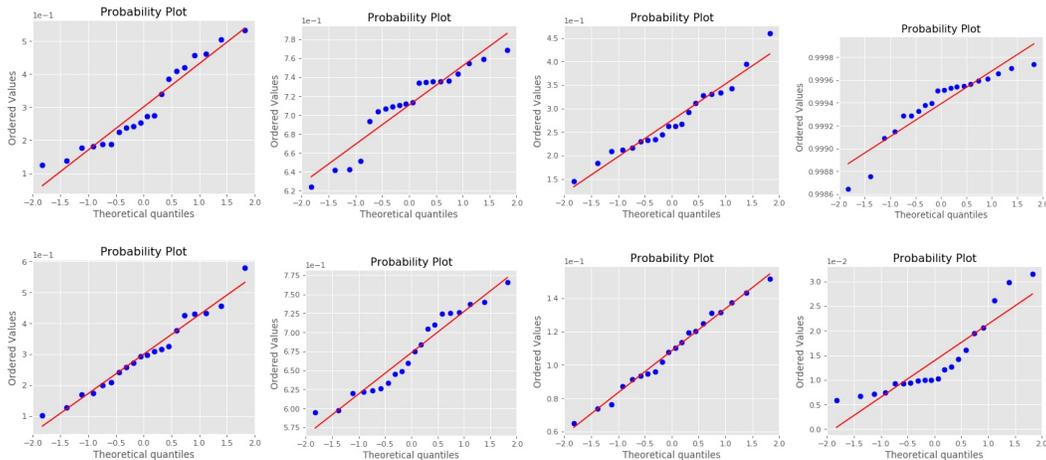


Figure 8: QQ Plot for output probabilities of VQA models: top row corresponds to the probability distributions of the class with the highest probability, and the bottom row corresponds to the probability distributions of the class with the second highest probability.

F.2 BOXPLOT FOR CIFAR-10

In this section, we visualize 5 most uncertain images for each dropout (only include Bernoulli, Concrete, and Contextual Bernoulli dropout for simplicity) leading to 15 images in total. The true images with the labels are on the left side and boxplots of probability distributions of different dropouts are on the right side. All models are trained on the original CIFAR-10 dataset. Among these 15 images, we observe that contextual dropout predicts the right answer if it is certain, and it is certain and predicts the right answer on many images that MC dropout or concrete dropout is uncertain about (e.g, many images in Figure 9-10). However, MC dropout or concrete dropout is uncertain about some easy examples (images in Figures 9-10) or certain on some wrong predictions (images in Figure 11). Moreover, on an image that all three methods have high uncertainty, concrete dropout often places a higher probability on the correct answer than the other two methods (images in Figure 11).

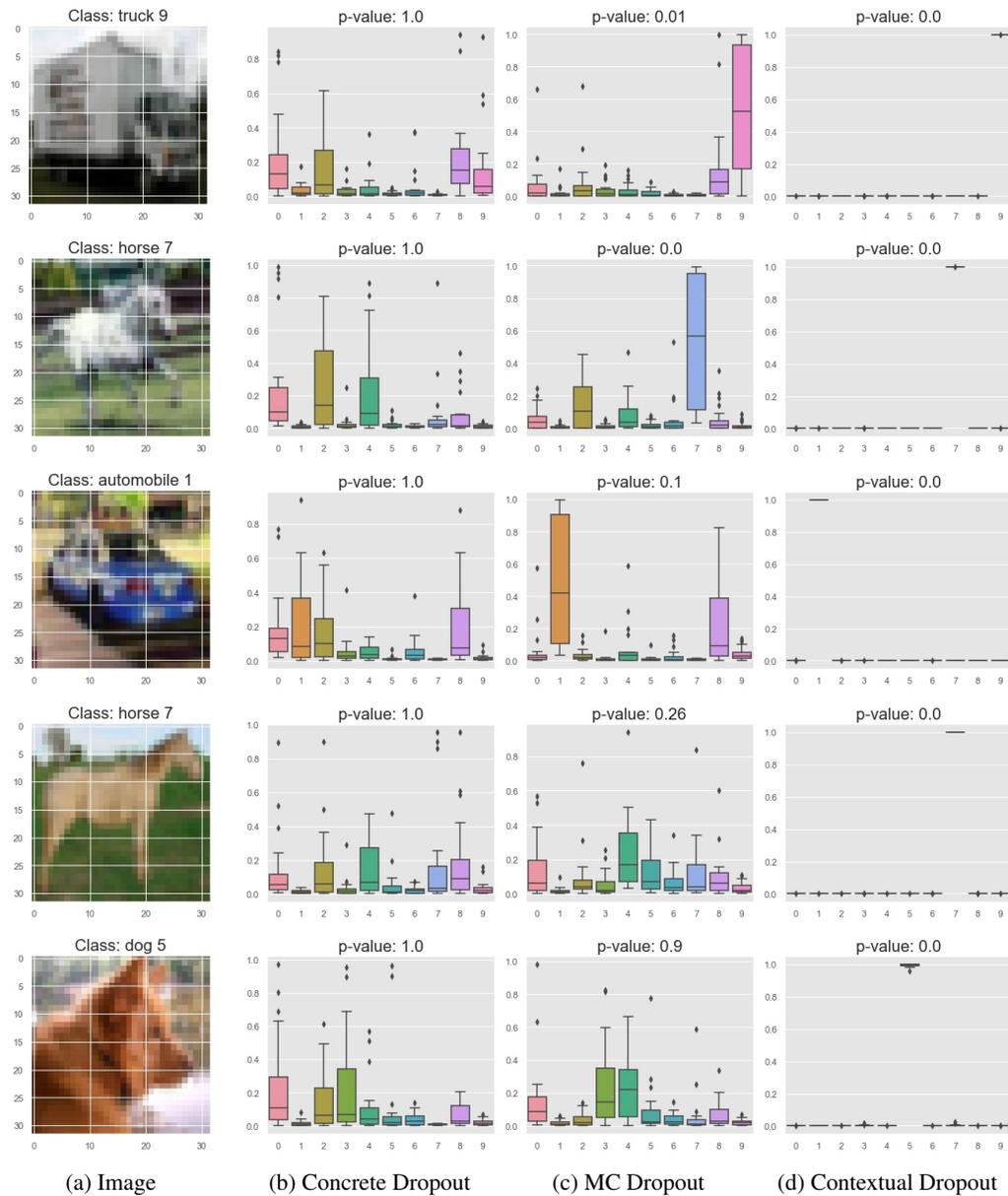


Figure 9: Visualization of probability outputs of different dropouts on CIFAR-10. 5 plots that **Concrete Dropout** is the most uncertain are presented. Number to class map: {0: airplane, 1: automobile, 2: bird, 3: cat, 4: deer, 5: dog, 6: frog, 7: horse, 8: ship, 9: truck.}

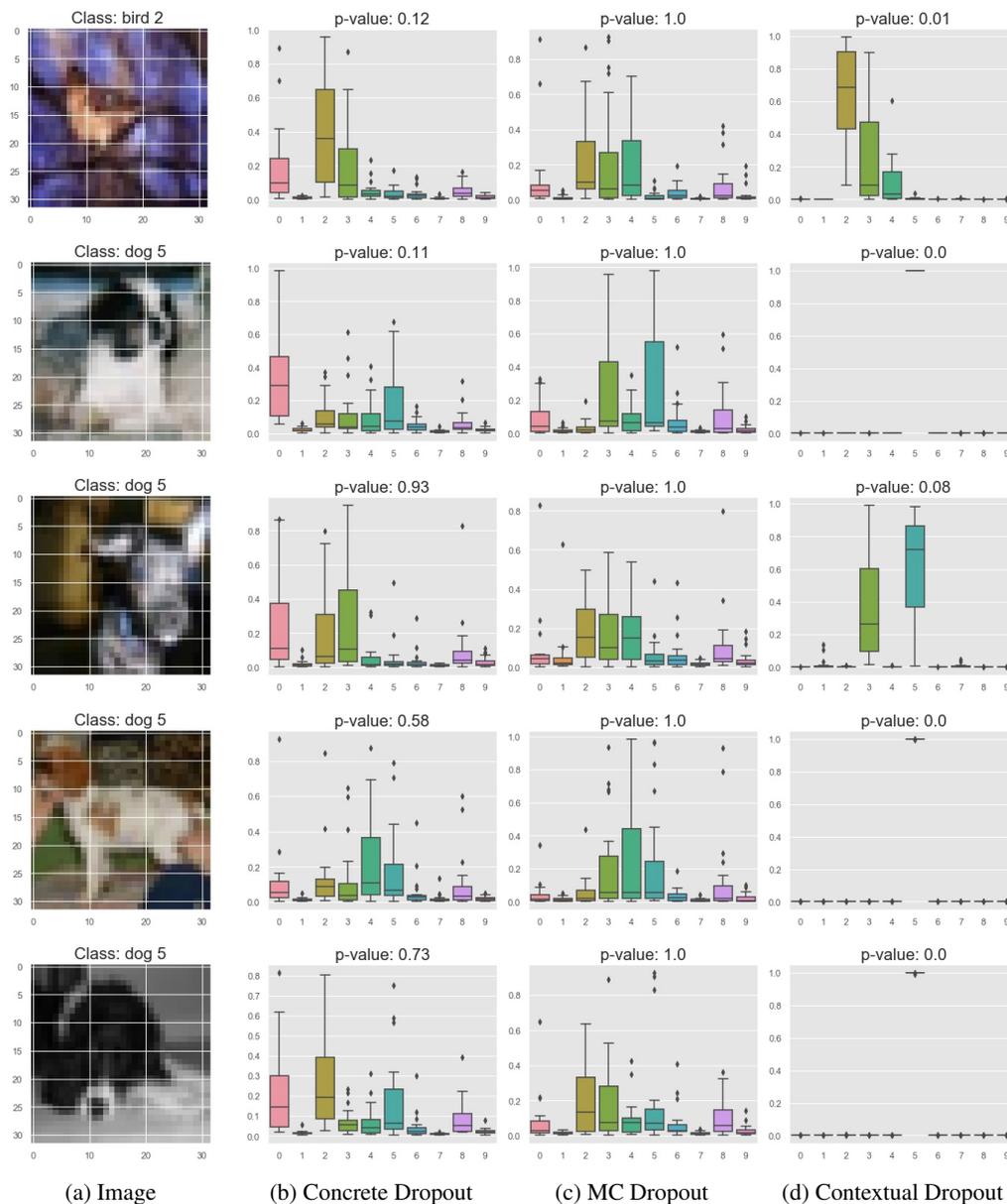


Figure 10: Visualization of probability outputs of different dropouts on CIFAR-10. 5 plots that **MC Dropout** is the most uncertain are presented. Number to class map: {0: airplane, 1: automobile, 2: bird, 3: cat, 4: deer, 5: dog, 6: frog, 7: horse, 8: ship, 9: truck.}

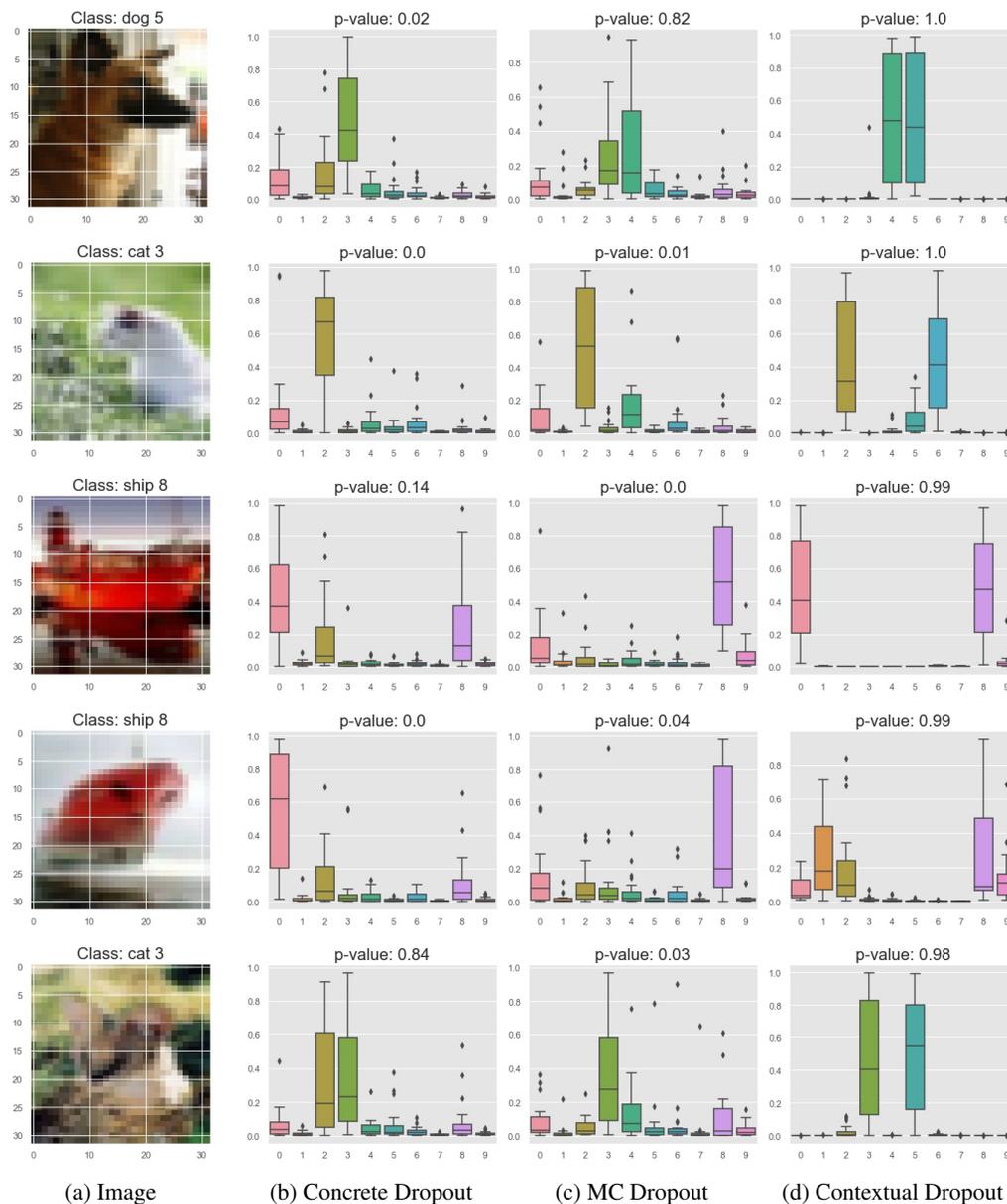


Figure 11: Visualization of probability outputs of different dropouts on CIFAR-10. 5 plots that **Contextual Dropout** is the most uncertain are presented. Number to class map: {0: airplane, 1: automobile, 2: bird, 3: cat, 4: deer, 5: dog, 6: frog, 7: horse, 8: ship, 9: truck.}

F.3 VISUALIZATION FOR VISUAL QUESTION ANSWERING

In Figures 12-15, we visualize some image-question pairs, along with the human annotations (for simplicity, we only show the different answers in the annotation set) and compare the predictions and uncertainty estimations of different dropouts (only include Bernoulli dropout, Concrete dropout, and contextual Bernoulli dropout) on the noisy data. We include 12 randomly selected image-question pairs, and 6 most uncertain image-question pairs for each dropout as challenging samples (30 in total). For each sample, we manually rank different methods by the general rule that accurate and certain is the most preferred, followed by accurate and uncertain, inaccurate and uncertain, and then inaccurate and certain. For each image-question pair, we rank three different dropouts based on their answers and p -values, and highlight the best performing one, the second best, and the worst with green, yellow, and red, respectively (tied ranks are allowed). As shown in the plots, overall contextual dropout is more conservative on its wrong predictions and more certain on its correct predictions than other methods for both randomly selected images and challenging images.

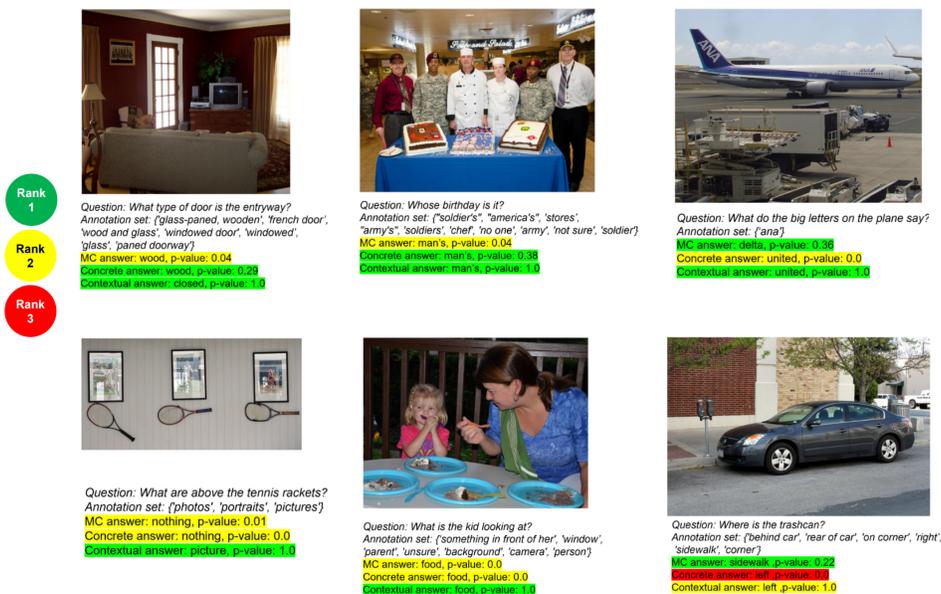


Figure 12: VQA visualization: 6 plots that **Contextual Dropout** is the most uncertain are presented.

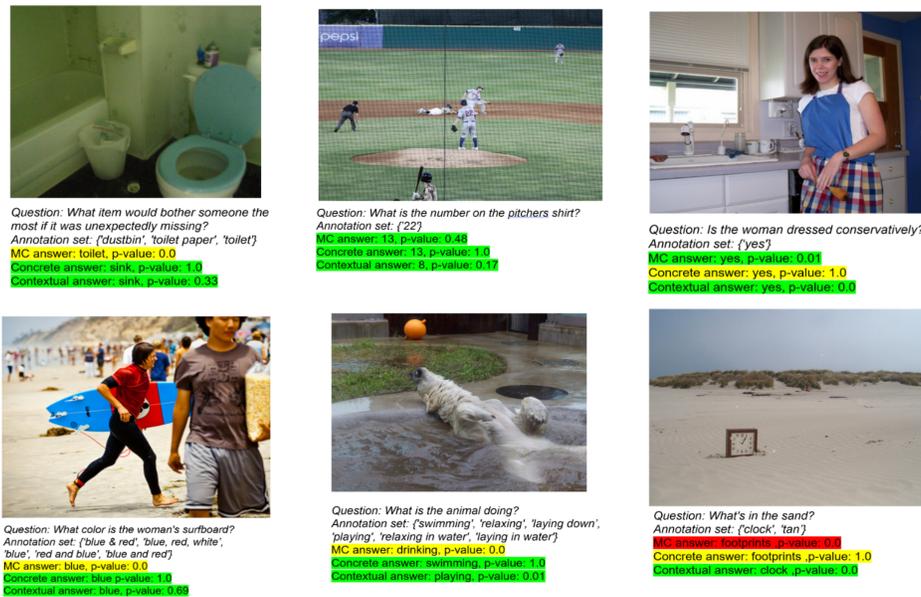


Figure 13: VQA visualization: 6 plots that **Concrete Dropout** is the most uncertain are presented.

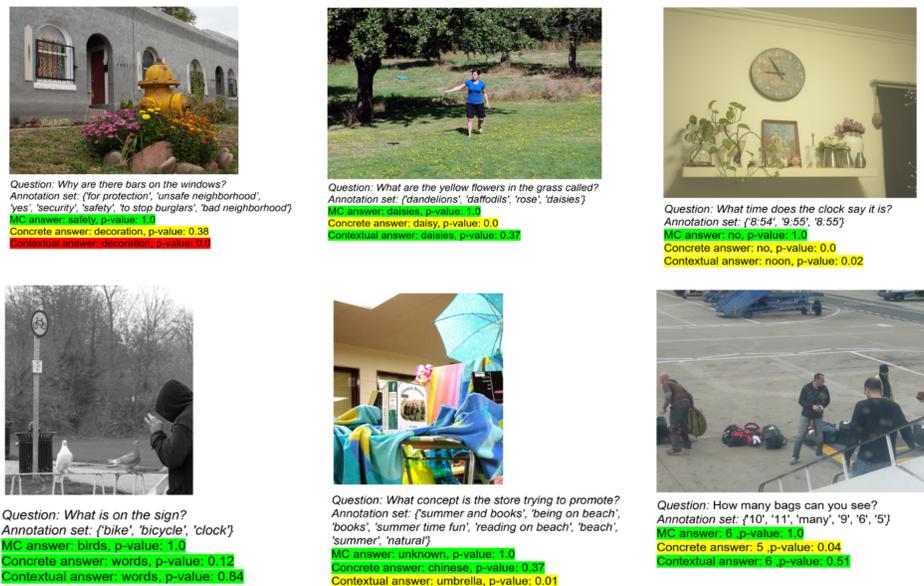


Figure 14: VQA visualization: 6 plots that **MC Dropout** is the most uncertain are presented.

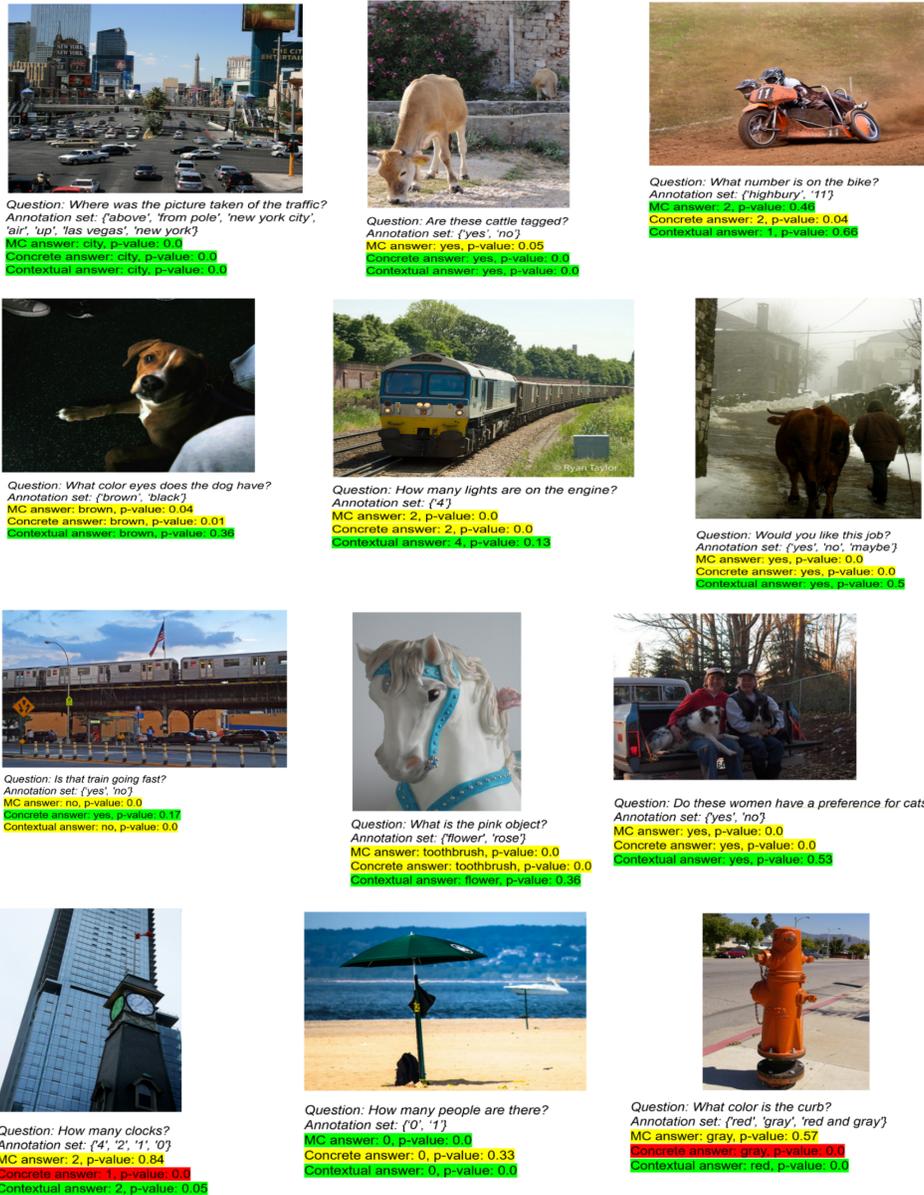


Figure 15: VQA visualization: 12 randomly selected plots are presented.