Robust Predictive Uncertainty for Neural Networks via Confidence Densities

Rob Cornish [1] **George Deligiannidis** [2] **Arnaud Doucet** [2]

Abstract

Neural networks are known to make incorrect predictions with high confidence when given inputs that are unlike their training set. This creates problems when a trained network is deployed on real-world data, since covariate shift away from the training set is usually inevitable. We propose *confidence densities* for explicitly controlling the predictions made by a network on out-of-sample inputs. Intuitively, our method makes confidence a finite resource for our network, which encourages making uniform predictions in regions away from the training data. Confidence densities may be used in conjunction with any off-the-shelf network architecture and can be trained using stochastic gradient descent. We show empirically that our method significantly improves out-of-sample robustness while maintaining the same classification accuracy across several datasets.

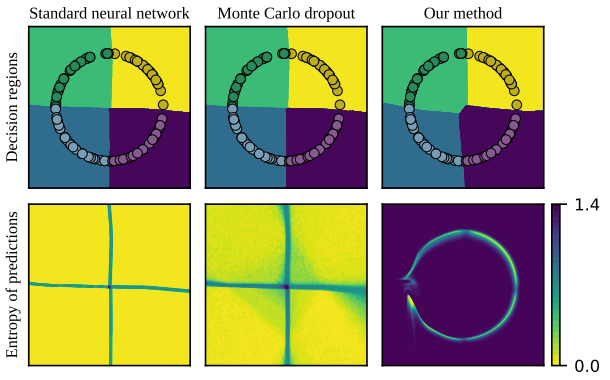

Figure 1. Comparison of predictive distributions learned for a simple classification problem with four classes. The top row shows the training set and the final decision regions, which are obtained from the class with highest predicted probability at each point in the space. The bottom row shows the entropy in nats of the predictive distribution at each point. All methods learn to classify the training set correctly, but our method additionally restricts its high-confidence predictions only to regions near the data.

1. Problem setting

We consider the task of learning a predictive distribution $p(y|x)$ given n i.i.d. data points $(x_i, y_i) \sim p(x, y)$, where the $x_i \in \mathcal{X}$ are data and the $y_i \in \{1, \dots, K\}$ are discrete labels. Here, given some x, $p(y|x)$ denotes the full distribution over labels. We will denote the probability that y specifically takes label k by $p(y = k|x)$.

At present, neural networks provide state-of-the-art classification performance for a wide variety of different $p(x, y)$. Standard methods use a given network architecture to parameterise a family of predictive distributions $q(y|x)$ (often by attaching a softmax output layer), and then try to choose $q(y|x)$ as close as possible to $p(y|x)$ as measured by some loss function. A widely used choice of loss function is the *cross entropy*, which is defined for our dataset as

$$\widehat{\mathcal{L}}[q(y|x)] := -\frac{1}{n}\sum_{i=1}^{n} \log q(y = y_i|x_i).$$

In the infinite data limit (i.e. as $n \to \infty$), this converges to

$$\mathcal{L}[q(y|x)] := \int \mathbb{D}_{\mathrm{KL}}[p(y|x)||q(y|x)]p(x)dx + \text{const} \quad (1)$$

where the constant does not depend on $q(y|x)$. Here $\mathbb{D}_{\mathrm{KL}}$ denotes the Kullback-Leibler divergence. It follows that cross entropy is a *proper scoring rule* (Gneiting & Raftery, 2007; Lakshminarayanan et al., 2017): given infinite data, its optimisers satisfy $q(y|x) = p(y|x)$ with $p(x)$-probability 1. See Section A in the Appendix for proofs.

With a large dataset, we can therefore expect that a well-trained $q(y|x)$ will make good predictions with high probability when $x \sim p(x)$. However, by itself, the cross entropy does not control the behaviour of $q(y|x)$ for *out-of-sample* x. Observe for instance that $\mathcal{L}[q(y|x)]$ does not depend at all on the values of $q(y|x)$ for which $p(x) = 0$. In practice,

[1]Department of Engineering Science, University of Oxford, Oxford, UK [2]Department of Statistics, University of Oxford, Oxford, UK. Correspondence to: Rob Cornish <rcornish@robots.ox.ac.uk>.

Presented at the ICML 2019 Workshop on Uncertainty and Robustness in Deep Learning.

when we use $\widehat{\mathcal{L}}[q(y|x)]$, this also holds if $p(x)$ is very small, since we will see very few such examples in our training set. As a result, $q(y|x)$ may make arbitrary predictions away from our training data. This has indeed been observed empirically. For instance, neural networks can make erroneous predictions with high confidence for data that has been adversarially perturbed (Szegedy et al., 2013), data that has been innocuously transformed (Engstrom et al., 2017), and data resembling random noise (Nguyen et al., 2014).

We seek to mitigate these problems by controlling the behaviour of $q(y|x)$ for out-of-sample x. In particular, if $p(x) = 0$, we will take the optimal $q(y|x)$ to be uniform. Thus our goal is effectively to learn

$$p^*(y|x) := \begin{cases} p(y|x), & p(x) > 0 \\ u(y), & p(x) = 0 \end{cases}$$

where $u(y)$ denotes the uniform distribution over K labels. This is more stringent than simply learning $p(y|x)$, which is undefined if $p(x) = 0$.

In general it is not clear $p^*(y|x)$ is always the best choice of target, since it might be preferred that our model generalise to certain x values that are outside the support of $p(x)$. A truly intelligent network trained on MNIST (LeCun & Cortes, 2010) might for example learn to classify MNIST digits whose black and white pixels have been reversed, even though it has never seen any such examples. However, we believe that in many applications – particularly safety-critical ones – $p^*(y|x)$ is a better choice of classifier than one that provides some generalisation but lacks other out-of-sample guarantees.

2. Our approach

To learn $p^*(y|x)$, we require a means to quantify how close a given $q(y|x)$ is to being uniform. A standard measure for this is *entropy* defined for a distribution on labels $q(y)$ as

$$\mathbb{H}[q(y)] := -\sum_{k=1}^{K} q(y=k) \log q(y=k).$$

Entropy takes values in $[0, \log K]$ and is largest when $q(y) = u(y)$. For our approach, it is more useful to work in terms of an affine transformation of the entropy that we term *confidence*, defined by

$$\mathbb{C}[q(y)] := \frac{\mathbb{H}[u(y)] - \mathbb{H}[q(y)]}{\mathbb{H}[u(y)]}.$$

Note that $\mathbb{C}[q(y)] \in [0, 1]$ with $\mathbb{C}[q(y)] = 0$ iff $q(y) = u(y)$, and $\mathbb{C}[q(y)] = 1$ iff $q(y)$ is Dirac.

Our idea is to control the *total confidence* of our classifier

$$\int \mathbb{C}[q(y|x)]dx. \tag{2}$$

Provided we can keep our predictions near the data fixed, it seems desirable for this quantity to be as small as possible. Intuitively, this has the effect of making confidence a finite resource, which encourages $q(y|x)$ trained with cross entropy to make high-confidence predictions only in regions near the data. More precisely, suppose the total confidence of $p^*(y|x)$ is finite. It then follows that $p^*(y|x)$ is the *unique* optimiser of

$$\min_{q(y|x) \in \mathcal{Q}^*} \int \mathbb{C}[q(y|x)]dx, \tag{3}$$

where $\mathcal{Q}^*$ denotes the set of minimisers of $\mathcal{L}[q(y|x)]$. See Proposition 3 in the Appendix for a proof. Thus our goal should ideally be to minimise the total confidence of $q(y|x)$ while not compromising its predictive accuracy. In practice this will not be possible exactly, and so our task is to balance these two concerns as best we can.

In general $p^*(y|x)$ may have infinite total confidence, for instance if it is Dirac on an unbounded set. In this case, *every* $q(y|x) \in \mathcal{Q}^*$ would optimise (3). However, if $\{x : p(x) > 0\}$ is bounded – which seems a reasonable assumption in many circumstances – then we have

$$\int \mathbb{C}[p^*(y|x)]dx = \int_{\{p(x)>0\}} \underbrace{\mathbb{C}[p(y|x)]}_{\leq 1} dx < \infty.$$

More generally, although we do not yet have theory to justify this, we conjecture that if we have access only to a finite sample from $p(x, y)$, then it makes sense to model $p^*(y|x)$ as having finite total confidence. We assume this is the case in what follows.

2.1. Confidence densities

In general the total confidence will be intractable to compute, and we therefore have no means to control it. We propose a parameterisation of $q(y|x)$ such that an upper bound to this quantity is always easily available. Observe that our assumption $\int \mathbb{C}[p^*(y|x)]dx < \infty$ simply means $\mathbb{C}[p^*(y|x)]$ is an unnormalised density. Our approach is to learn this unnormalised density jointly as part of our model. Specifically, we will take

$$q(y|x) := \pi[\nu(y|x)|c(x)],$$

where:

- $\nu(y|x)$ is a *base classifier* that outputs a predictive distribution over labels for each x;
- $c(x) : \mathcal{X} \to [0, 1]$ is a *confidence density* satisfying

$$\int c(x)dx < \infty;$$

- $\pi[\nu(y)|c]$ projects a distribution on labels $\nu(y)$ and $c \in [0,1]$ onto another distribution on labels whose confidence is at most c. Precisely,

$$\mathbb{C}[\pi[\nu(y)|c]] \leq c.$$

We also assume $\pi[\nu(y)|c] = \nu(y)$ if $\mathbb{C}[\nu(y|x)] \leq c$.

By definition, this yields

$$\int \mathbb{C}[q(y|x)]dx \leq \int c(x)dx < \infty.$$

Moreover, we do not necessarily lose expressiveness through this construction: any $p^*(y|x)$ with finite total confidence is obtained simply by taking $c(x) = \mathbb{C}[p^*(y|x)]$ and $\nu(y|x) = p(y|x)$.

2.2. A practical implementation

We now provide an instance of this setup that is suitable for use in practice. Our choice of $\nu(y|x)$ is essentially unrestricted: we can use for example any neural network architecture with K softmax outputs. For π we will use

$$\pi[\nu(y)|c] := \gamma(x)\nu(y) + (1-\gamma(x))u(y)$$

where

$$\gamma(x) := \min\left(1, \frac{c}{\mathbb{C}[\nu(y)]}\right).$$

By the convexity of $\mathbb{C}$, which follows from concavity of $\mathbb{H}$,

$$\mathbb{C}[\pi[\nu(y)|c]] \leq \underbrace{\gamma\mathbb{C}[\nu(y)]}_{\leq c} + (1-\gamma)\underbrace{\mathbb{C}[u(y)]}_{=0} \leq c.$$

It is also clear that $\pi[\nu(y)|c] = \nu(y)$ if $\mathbb{C}[\nu(y)] \leq c$.

For the confidence density we propose taking

$$c(x) := \sigma(zw(x)),$$

where $w(x)$ is a properly normalised density, $z \in [0,\infty)$, and $\sigma(t) : [0,\infty) \to [0,1]$ satisfies $\sigma(t) \leq t$. In the following we specifically set

$$\sigma(t) := 1 - \exp(-t).$$

It is straightforward to see that these choices yield the desired properties for $c(x)$. In our experiments we take $w(x)$ to be a MAF (Papamakarios et al., 2017), but we could use any density estimator that allows exact computation of likelihoods, such as a Real NVP (Dinh et al., 2016) or Glow (Kingma & Dhariwal, 2018).

This construction entails that

$$\int c(x)dx = \int \sigma(zw(x))dx \leq z\int w(x)dx = z.$$

Thus, not only do we ensure the total confidence is finite, but we also have a mechanism for controlling its value in a smooth way through z. In light of (3), it seems reasonable that z should be made as small as possible without increasing the loss too much. Formulating a more principled approach is left for future work.

In general, this form of $q(y|x)$ does entail some loss in expressiveness. In particular, in order to allow the possibility $q(y|x) = p^*(y|x)$, it must be the case that

$$\int_{\{p(x)>0\}} -\log(1-\mathbb{C}[p(y|x)])dx < \infty. \tag{4}$$

To see why, note that if $q(y|x) = p^*(y|x)$, then we must have $c(x) \geq \mathbb{C}[p^*(y|x)]$ for all x. This means

$$\begin{aligned}\int w(x)dx &\geq \int z^{-1}\sigma^{-1}(\mathbb{C}[p^*(y|x)])dx \\ &= z^{-1}\int -\log(1-\mathbb{C}[p^*(y|x)])dx,\end{aligned}$$

which in turn yields (4) since $w(x)$ is a density. (4) holds for instance if $\mathbb{C}[p(y|x)]$ is bounded away from 1. This does not seem a severe assumption and in practice we have not found it to be a limitation, with $q(y|x)$ of this form able to achieve competitive performance on a variety of datasets.

2.2.1. Loss function

Assuming $w(x)$ and $\nu(y|x)$ are differentiable, we would ideally learn their parameters end-to-end by optimising $\widehat{\mathcal{L}}[q(y|x)]$ via gradient descent. However, we found empirically this causes vanishing gradients as the dimension of the data grows. In particular, observe that, if $\sigma(zw(x)) < \mathbb{C}[\nu(y|x)]$, then, for a given training example (x_i, y_i),

$$\nabla_w \log q(y = y_i|x_i) = P(x_i, y_i)\nabla_w \log w(x_i)$$

where

$$P(x_i, y_i) := \frac{(\nu(y = y_i|x_i) - K^{-1})zw(x_i)\exp(-zw(x_i))}{q(y = y_i|x_i)\mathbb{C}[\nu(y = y_i|x_i)]}.$$

Since $t\exp(-t) \to 0$ both as $t \to 0$ and as $t \to \infty$, there is consequently only a relatively small range of values of $zw(x_i)$ for which the cross-entropy produces a nonnegligible gradient with respect to the parameters of $w(x)$. In practice, in high dimensions, a typical density estimator such as a MAF will produce outputs that vary over very many orders of magnitude, which results in extremely weak gradients during training for most data points.

We address this issue as follows. As noted above, an optimal choice of $q(y|x)$ of our proposed form has $\nu(y|x) = p(y|x)$. Accordingly, we first train $\nu(y|x)$ independently to minimise $\mathcal{L}[\nu(y|x)]$. Then we train $w(x)$ using the loss function

$$\widehat{\mathcal{L}}^+[q(y|x)] := -\frac{1}{n}\sum_{i=1}^{n}(\nu(y = y_i|x_i) - K^{-1})\log w(x_i)$$

with $\nu(y|x)$ held fixed. This yields gradient terms

$$(\nu(y = y_i|x_i) - K^{-1})\nabla_w \log w(x_i),$$

which from (5) we see have the correct direction, at least individually. We found this yields a significantly stronger gradient suitable for use in practice. It also has the benefit of not requiring a choice of z at training time.

3. Experiments

We now present the results of applying our method to several datasets. In all cases we trained $q(y|x)$ using the methodology described in Section 2.2.1, although for the synthetic dataset we obtained essentially the same result when optimising $\widehat{\mathcal{L}}[q(y|x)]$ end-to-end. For $\nu(y|x)$ we used a multilayer perceptron with ReLU nonlinearities and softmax outputs. For $w(x)$ we used a MAF (Papamakarios et al., 2017) with 5 layers and $\tanh$ nonlinearities. After training, we chose z to be as small as possible without significantly reducing the average confidence of $q(y|x)$ on a held-out a validation set.

As a baseline, for each dataset we also trained $\nu(y|x)$ separately by minimising $\widehat{\mathcal{L}}[\nu(y|x)]$, and evaluated its predictions with and without Monte Carlo dropout (MC dropout) (Gal & Ghahramani, 2015). The label distributions for the latter were taken as the average of 100 dropout samples.

3.1. Synthetic data

We evaluated our method using the toy dataset shown in Figure 1. Here we sampled 200 points x_i uniformly on the unit circle, with y_i determined by quadrant. $\nu(y|x)$ had a single hidden layer with 40 neurons, and a dropout layer with rate 0.2 after the nonlinearity. At test time, for MC dropout, we made increased this rate to 0.5 Each layer in $w(x)$ had 64 hidden units. We trained via gradient descent with a learning rate of 0.01 for 20000 steps using the full training set at each iteration. We took $\log z$ to be 1.3.

As can be seen, all methods learn essentially the same predictive distributions for in-sample data. However, the baselines make confident predictions away from the unit circle, where there is no data, while our predictor is essentially uniform.

3.2. MNIST, Fashion MNIST, and notMNIST

We also applied our method to the MNIST (LeCun & Cortes, 2010) and Fashion MNIST (Xiao et al., 2017) datasets. Here each $\nu(y|x)$ had two hidden layers each with 128 neurons, with rate 0.2 dropout layers after each nonlinearity. Each layer in $w(x)$ had 1024 hidden neurons. We split off 10000 training examples to use as a validation set, and stopped training after 50 epochs of no improvement on the best validation score seen so far. For our method, $\widehat{\mathcal{L}}^{+}[q(y|x)]$ was used for this validation score when training $w(x)$. At test time in both cases we took $\log z = 1550$.

Table 1. Comparison of test accuracy and average confidence for a standard neural network (NN), Monte Carlo dropout (MCD) with dropout rates 0.2 and 0.5, and a confidence density predictor (CD). The top four rows were obtained after training on MNIST, and the bottom four after training on Fashion MNIST (FMNIST). Accuracies were computed as the fraction of labels that were correctly predicted as most likely. Bold text indicates the best average confidences for out-of-sample inputs (lower is better). Error bars were negligible.

Method	**MNIST**		**FMNIST**		**notMNIST**	
	Acc.	Conf.	Acc.	Conf.	Acc.	Conf.
NN	.973	.979	.068	.603	.101	.816
MCD (0.2)	.974	.930	.068	.510	.099	.624
MCD (0.5)	.971	.728	.069	.333	.095	.370
CD	.973	.967	.068	**.015**	.101	**.000**
NN	.097	.646	.883	.887	.080	.825
MCD (0.2)	.096	.533	.884	.853	.082	.664
MCD (0.5)	.096	.348	.880	.709	.086	.431
CD	.097	**.096**	.883	.848	.080	**.000**

Since our $q(y|x)$ is a convex combination of $\nu(y|x)$ and $u(y)$, it is easily seen that

$$\arg\max_{1\leq k\leq K} q(y = k|x) = \arg\max_{1\leq k\leq K} \nu(y = k|x)$$

for any x. In this sense our method always maintains the same predictive accuracy as the base classifier $\nu(y|x)$. We found that MC dropout also yielded essentially the same accuracies. Table 1 contains a summary of these results.. Observe that in each instance the confidence density predictor correctly makes close to uniform predictions for the two out-of-sample datasets, while both the standard neural network and MC dropout are overconfident. Moreover, this is achieved without a significant reduction in confidence for the predictions made for in-sample data.

4. Related work

In addition to MC dropout, various other methods for improving the uncertainty estimates of neural networks have been proposed, including (Lakshminarayanan et al., 2017; Hafner et al., 2018; Nalisnick et al., 2019). However, we are not aware of any method that explicitly controls the total predictive confidence in the way we do. These alternative methods may in fact prove complimentary to our own. For instance, using an ensemble of confidence density classifiers might further improve performance as it did for Lakshminarayanan et al. (2017). Exploring this possibility and properly benchmarking against these other approaches is left for future work.

Acknowledgements

We are very grateful for the many helpful comments of Adam Golinski and Anthony Caterini throughout the creation of this work. Rob Cornish is supported by the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines & Systems (EP/L015897/1) and NVIDIA. Arnaud Doucet is partially supported by the U.S. Army Research Laboratory, the U. S. Army Research Office, and by the U.K. Ministry of Defence (MoD) and the U.K. EPSRC under grant numbers EP/R013616/1 and EP/R034710/1.

References

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *arXiv e-prints*, art. arXiv:1605.08803, May 2016.

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations. *arXiv e-prints*, art. arXiv:1712.02779, Dec 2017.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *arXiv e-prints*, art. arXiv:1506.02142, Jun 2015.

Gneiting, T. and Raftery, A. E. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437. URL https://doi.org/10.1198/016214506000001437.

Hafner, D., Tran, D., Lillicrap, T., Irpan, A., and Davidson, J. Reliable Uncertainty Estimates in Deep Neural Networks using Noise Contrastive Priors. *arXiv e-prints*, art. arXiv:1807.09289, Jul 2018.

Kingma, D. P. and Dhariwal, P. Glow: Generative Flow with Invertible 1x1 Convolutions. *arXiv e-prints*, art. arXiv:1807.03039, Jul 2018.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6402–6413. Curran Associates, Inc., 2017.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Nalisnick, E., Matsukawa, A., Whye Teh, Y., Gorur, D., and Lakshminarayanan, B. Hybrid Models with Deep and Invertible Features. *arXiv e-prints*, art. arXiv:1902.02767, Feb 2019.

Nguyen, A., Yosinski, J., and Clune, J. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. *arXiv e-prints*, art. arXiv:1412.1897, Dec 2014.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked Autoregressive Flow for Density Estimation. *arXiv e-prints*, art. arXiv:1705.07057, May 2017.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv e-prints*, art. arXiv:1312.6199, Dec 2013.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv e-prints*, art. arXiv:1708.07747, Aug 2017.

A. Proofs

Proposition 1. *Under appropriate integrability conditions, given i.i.d.* $(x_i, y_i) \sim p(x, y)$

$$-\frac{1}{n}\sum_{i=1}^{n} \log q(y = y_i|x_i) \to \mathcal{L}[q(y|x)]$$

$p(x, y)$*-almost surely as* $n \to \infty$.

Proof. By the law of large numbers,

$$-\frac{1}{n}\sum_{i=1}^{n} \log q(y = y_i|x_i) \to -\int \sum_{k=1}^{K} p(x, y = k) \log q(y = k|x) dx, \quad (5)$$

so it just remains to show that the right-hand side has the desired form (1). Now

$$\begin{aligned} -\sum_{k=1}^{K} p(x, y = k) \log q(y = k|x) &= -p(x) \sum_{k=1}^{K} p(y = k|x) \log q(y = k|x) \\ &= p(x)(\mathbb{D}_{\mathrm{KL}}[p(y|x)||q(y|x)] + \mathbb{H}[p(y|x)]), \end{aligned}$$

where $\mathbb{H}$ denotes the entropy. Consequently, the right-hand side of (5) may be written

$$\int \mathbb{D}_{\mathrm{KL}}[p(y|x)||q(y|x)]p(x)dx + \int \mathbb{H}[p(y|x)]p(x)dx,$$

which gives the result. □

Proposition 2. *$q(y|x)$ minimises $\mathcal{L}[q(y|x)]$ iff $q(y|x) = p(y|x)$ with $p(x)$-probability* 1.

Proof. Minimising $\mathcal{L}[q(y|x)]$ is equivalent to minimising

$$\int \mathbb{D}_{\mathrm{KL}}[p(y|x)||q(y|x)]p(x)dx.$$

The result is then a straightforward consequence of the fact that $\mathbb{D}_{\mathrm{KL}}[p(y|x)||q(y|x)] \geq 0$ with equality iff $q(y|x) = p(y|x)$. □

Proposition 3. *If*

$$\int \mathbb{C}[p^*(y|x)]dx < \infty$$

and

$$\mathcal{Q}^* = \arg\min_{q(y|x)} \mathcal{L}[q(y|x)],$$

then $p^*(y|x)$ is the unique optimiser of

$$\min_{q(y|x)\in\mathcal{Q}^*} \int \mathbb{C}[q(y|x)]dx \tag{6}$$

up to a set of dx-measure 0*.*

Proof. Suppose $q(y|x) \in \mathcal{Q}^*$. By Proposition (2), we have $q(y|x) = p(y|x)$ for dx-almost every x with $p(x) > 0$, and so

$$\begin{aligned}&\int \mathbb{C}[q(y|x)]dx \\ &= \int_{\{p(x)>0\}} \mathbb{C}[p(y|x)]dx + \int_{\{p(x)=0\}} \mathbb{C}[q(y|x)]dx.\end{aligned}$$

Since

$$\int_{\{p(x)>0\}} \mathbb{C}[p(y|x)]dx = \int \mathbb{C}[p^*(y|x)]dx < \infty,$$

the optimisation (6) is equivalent to

$$\min_{q(y|x)\in\mathcal{Q}^*} \int_{\{p(x)=0\}} \mathbb{C}[q(y|x)]dx.$$

Since $\mathbb{C}$ is nonnegative, this integral is 0 iff $\mathbb{C}[q(y|x)] = 0$ for dx-almost every x with $p(x) = 0$. But $\mathbb{C}[q(y|x)] = 0$ iff $q(y|x) = u(y)$. It follows that any minimiser of (6) must be equal to $p^*(y|x)$ for dx-almost every x. □