Lightweight Probabilistic Deep Networks
– Supplemental Material –

Jochen Gast Stefan Roth
Department of Computer Science, TU Darmstadt

In this supplemental material we derive the recipe to create uncertainty propagating layers, give additional details on the various types of layers used in the main paper, visualize the effect of uncertainty prediction, and illustrate its benefits with example classification and regression results.

A. Assumed Density Filtering

We now derive the necessary equations to show that our framework indeed corresponds to ADF applied to the joint density of neural network activations. Let us briefly recall the assumptions of our main paper. The true joint density of network activations in a deterministic network with Gaussian independent inputs is given by

$$p(\mathbf{z}^{(0:l)}) = p(\mathbf{z}^{(0)}) \prod_{i=1}^{l} p(\mathbf{z}^{(i)} \,|\, \mathbf{z}^{(i-1)}) \tag{22a}$$

with

$$p(\mathbf{z}^{(i)} \,|\, \mathbf{z}^{(i-1)}) = \delta\big[\mathbf{z}^{(i)} - \mathbf{f}^{(i)}(\mathbf{z}^{(i-1)})\big] \tag{22b}$$

$$p(\mathbf{z}^{(0)}) = \prod_{j} \mathcal{N}\left(z_j^{(0)} \,|\, x_j, \sigma_n^2\right). \tag{22c}$$

The approximate joint density is assumed to be

$$q(\mathbf{z}^{(0:l)}) = q(\mathbf{z}^{(0)}) \prod_{i=1}^{l} q(\mathbf{z}^{(i)}), \qquad q\big(\mathbf{z}^{(i)}\big) = \prod_{j} \mathcal{N}\left(z_j^{(i)} \,|\, \mu_j^{(i)}, v_j^{(i)}\right). \tag{23}$$

As explained in the main paper, we initialize the first factor of our approximation by setting $q(\mathbf{z}^{(0)}) = p(\mathbf{z}^{(0)})$. This is possible since we assume Gaussian input noise. Afterwards, ADF performs iterative approximations by minimizing

$$\underset{\tilde{q}(\mathbf{z}^{(0:i)})}{\arg\min} \quad \mathrm{KL}(\tilde{p}(\mathbf{z}^{(0:i)}) \parallel \tilde{q}(\mathbf{z}^{(0:i)})), \tag{24}$$

where the true posterior at each iteration consists of the newest factor and the previous approximating factors

$$\tilde{p}(\mathbf{z}^{(0:i)}) = p(\mathbf{z}^{(i)} \,|\, \mathbf{z}^{(i-1)}) \prod_{j=0}^{i-1} q(\mathbf{z}^{(j)}). \tag{25}$$

Updates are then performed for the consecutive iterations (or layers) $i = 1, \ldots, l$. For the Gaussian assumption (Eq. 23), the solution to Eq. (24) equals to matching the first-order and second-order moments of $\tilde{p}(\mathbf{z}^{(0:i)})$ and $\tilde{q}(\mathbf{z}^{(0:i)})$.

We now show that for neural network activations as modeled by Eqs. (22a) to (22c), solving Eq. (24) exactly yields our recipe from Eqs. (9a) and (9b) of the main paper for creating uncertainty propagating layers. To show this, the following two properties of Dirac delta distributions are quite useful:

$$\int_{-\infty}^{\infty} \delta[x - y] g(x) \,\mathrm{d}x = g(y) \quad \text{and} \quad \int_{-\infty}^{\infty} \delta[x - y] \,\mathrm{d}x = 1. \tag{26}$$

Also note that we will factorize the approximate posterior by removing an activation variable z_k from the distribution and explicitly represent it as a separate factor, *i.e.* $q(\mathbf{z}) = q(\mathbf{z}_{\setminus k})\, q(z_k)$, where $q(\mathbf{z}_{\setminus k})$ corresponds to the joint density of all factors excluding z_k.

We continue to derive the first order moment of $\tilde{p}(\mathbf{z}^{(0:i)})$, which will be done in two steps. First, we derive the moment of activation variables z_k of all layers excluding the last layer, *i.e.* for z_k that are an element of $\mathbf{z}^{(0:i-1)}$. Afterwards, we take a look at activation variables solely contained in the last layer, *i.e.* for z_k elements of $\mathbf{z}^{(i)}$.

For activation variables z_k not contained in the last layer (*i.e.* part of $\mathbf{z}^{(0:i-1)}$), we have the first moment

$$\mathbb{E}_{\tilde{p}}[z_k] = \int_{\mathbf{Z}} \tilde{p}(\mathbf{z}^{(0:i)}) z_k \,\mathrm{d}\mathbf{z} = \int_{-\infty}^{\infty} \int_{\mathbf{Z}_{\setminus k}} \tilde{p}(\mathbf{z}^{(0:i)}) z_k \,\mathrm{d}\mathbf{z}_{\setminus k} \,\mathrm{d}z_k \tag{27a}$$

$$= \int_{-\infty}^{\infty} \int_{\mathbf{Z}_{\setminus k}^{(0:i-1)}} \int_{\mathbf{Z}^{(i)}} \Big(\delta[\mathbf{z}^{(i)} - \mathbf{f}^{(i)}(\mathbf{z}^{(i-1)})] q(z_k) q(\mathbf{z}_{\setminus k}^{(0:i-1)}) z_k\Big) \,\mathrm{d}\mathbf{z}^{(i)} \,\mathrm{d}\mathbf{z}_{\setminus k}^{(0:i-1)} \,\mathrm{d}z_k \tag{27b}$$

$$= \int_{-\infty}^{\infty} q(z_k) z_k \Big(\int_{\mathbf{Z}_{\setminus k}^{(0:i-1)}} q(\mathbf{z}_{\setminus k}^{(0:i-1)}) \Big(\int_{\mathbf{Z}^{(i)}} \delta[\mathbf{z}^{(i)} - \mathbf{f}^{(i)}(\mathbf{z}^{(i-1)})] \,\mathrm{d}\mathbf{z}^{(i)} \Big) \,\mathrm{d}\mathbf{z}_{\setminus k}^{(0:i-1)} \Big) \,\mathrm{d}z_k \tag{27c}$$

$$= \int_{-\infty}^{\infty} q(z_k) z_k \Big(\int_{\mathbf{Z}_{\setminus k}^{(0:i-1)}} q(\mathbf{z}_{\setminus k}^{(0:i-1)}) \,\mathrm{d}\mathbf{z}_{\setminus k}^{(0:i-1)} \Big) \,\mathrm{d}z_k = \int_{-\infty}^{\infty} q(z_k) z_k \,\mathrm{d}z_k = \mathbb{E}_{q(z_k)}[z_k]. \tag{27d}$$

Hence, for all layers except for the i^{th} layer, the first moments remain unchanged after the update. For activation variables z_k solely contained in the last layer $\mathbf{z}^{(i)}$, we have the first moment

$$\mathbb{E}_{\tilde{p}}[z_k] = \int_{\mathbf{Z}} \tilde{p}(\mathbf{z}^{(0:i)}) z_k \,\mathrm{d}\mathbf{z} = \int_{\mathbf{Z}_{\setminus k}} \int_{-\infty}^{\infty} \tilde{p}(\mathbf{z}^{(0:i)}) z_k \,\mathrm{d}z_k \,\mathrm{d}\mathbf{z}_{\setminus k} \tag{28a}$$

$$= \int_{\mathbf{Z}^{(0:i-1)}} \int_{\mathbf{Z}_{\setminus k}^{(i)}} \int_{-\infty}^{\infty} \Big(\delta[\mathbf{z}^{(i)} - \mathbf{f}^{(i)}(\mathbf{z}^{(i-1)})] q(\mathbf{z}^{(0:i-1)}) z_k\Big) \,\mathrm{d}z_k \,\mathrm{d}\mathbf{z}_{\setminus k}^{(i)} \,\mathrm{d}\mathbf{z}^{(0:i-1)} \tag{28b}$$

$$= \int_{\mathbf{Z}^{(0:i-1)}} q(\mathbf{z}^{(0:i-1)}) \Big(\int_{\mathbf{Z}_{\setminus k}^{(i)}} \int_{-\infty}^{\infty} \Big(\delta[\mathbf{z}^{(i)} - \mathbf{f}^{(i)}(\mathbf{z}^{(i-1)})] z_k\Big) \,\mathrm{d}z_k \,\mathrm{d}\mathbf{z}_{\setminus k}^{(i)} \Big) \,\mathrm{d}\mathbf{z}^{(0:i-1)} \tag{28c}$$

$$= \int_{\mathbf{Z}^{(0:i-1)}} q(\mathbf{z}^{(0:i-1)}) \Big(\int_{\mathbf{Z}_{\setminus k}^{(i)}} \int_{-\infty}^{\infty} \Big(\delta[z_k - f_k^{(i)}(\mathbf{z}^{(i-1)})] \delta[\mathbf{z}_{\setminus k}^{(i)} - \mathbf{f}_{\setminus k}^{(i)}(\mathbf{z}^{(i-1)})] z_k\Big) \,\mathrm{d}z_k \,\mathrm{d}\mathbf{z}_{\setminus k}^{(i)} \Big) \,\mathrm{d}\mathbf{z}^{(0:i-1)} \tag{28d}$$

$$= \int_{\mathbf{Z}^{(0:i-1)}} q(\mathbf{z}^{(0:i-1)}) \Big(\int_{-\infty}^{\infty} \delta[z_k - f_k^{(i)}(\mathbf{z}^{(i-1)})] z_k \,\mathrm{d}z_k \Big) \Big(\int_{\mathbf{Z}_{\setminus k}^{(i)}} \delta[\mathbf{z}_{\setminus k}^{(i)} - \mathbf{f}_{\setminus k}^{(i)}(\mathbf{z}^{(i-1)})] \,\mathrm{d}\mathbf{z}_{\setminus k}^{(i)} \Big) \,\mathrm{d}\mathbf{z}^{(0:i-1)} \tag{28e}$$

$$= \int_{\mathbf{Z}^{(0:i-1)}} q(\mathbf{z}^{(0:i-1)}) f_k^{(i)}(\mathbf{z}^{(i-1)}) \,\mathrm{d}\mathbf{z}^{(0:i-1)} = \int_{\mathbf{Z}^{(i-1)}} q(\mathbf{z}^{(i-1)}) f_k^{(i)}(\mathbf{z}^{(i-1)}) \,\mathrm{d}\mathbf{z}^{(i-1)} = \mathbb{E}_{q^{(i-1)}}[f_k^{(i)}(\mathbf{z}^{(i-1)})], \tag{28f}$$

which exactly corresponds to our recipe in Eq. (9a). Note that by replacing z_k with z_k^2 in the derivations above, it is easy to obtain similar results for the second order moments:

$$\forall z_k \text{ elements of } \mathbf{z}^{(0:i-1)}: \quad \mathbb{E}_{\tilde{p}}[z_k^2] = \mathbb{E}_{q(z_k)}[z_k^2] \tag{29a}$$

$$\forall z_k \text{ elements of } \mathbf{z}^{(i)}: \quad \mathbb{E}_{\tilde{p}}[z_k^2] = \mathbb{E}_{q^{(i-1)}}[f_k^2(\mathbf{z}^{(i-1)})], \tag{29b}$$

which yields the variance update of the recipe Eq. (9b).

B. Uncertainty Propagation Layers

We now describe the details of various layer implementations. Recall the general recipe as given in the main paper, which we just derived as

$$\boldsymbol{\mu}_z^{(i)} = \mathbb{E}_{q(\mathbf{z}^{(i-1)})}\Big[\mathbf{f}^{(i)}(\mathbf{z}^{(i-1)}; \boldsymbol{\theta}^{(i)})\Big] \tag{30a}$$

$$\mathbf{v}_z^{(i)} = \mathbb{V}_{q(\mathbf{z}^{(i-1)})}\Big[\mathbf{f}^{(i)}(\mathbf{z}^{(i-1)}; \boldsymbol{\theta}^{(i)})\Big]. \tag{30b}$$

Linear layers. As explained in the main paper, fully connected layers, convolutions, and deconvolutions can all be formalized as linear functions

$$\mathbf{z}^{(i)} = W\mathbf{z}^{(i-1)} + \mathbf{b}. \tag{31}$$

By the linearity of the expectation, the uncertainty propagation layer is consequently given by

$$\boldsymbol{\mu}_z^{(i)} = W\boldsymbol{\mu}_z^{(i-1)} + \mathbf{b} \tag{32a}$$

$$\mathbf{v}_z^{(i)} = (W \circ W)\mathbf{v}_z^{(i-1)}. \tag{32b}$$

(Global) average pooling. The uncertainty propagation layer for average pooling

$$z^{(i)} = \text{mean}\left(\mathbf{z}^{(i-1)}\right) \tag{33}$$

can be derived by formalizing it as a linear layer, where weights are chosen such that they correspond to the averaging operator. In practice, we can implement the layer via the following equations:

$$\mu_z^{(i)} = \text{mean}(\boldsymbol{\mu}_z^{(i-1)}) \tag{34a}$$

$$v_z^{(i)} = \frac{1}{n}\,\text{mean}(\mathbf{v}_z^{(i-1)}), \tag{34b}$$

where for Eq. (34b) we rely on the fact that the mean operator itself already multiplies $1/n$ to the variances. Here, n is the number of elements of $\mathbf{z}^{(i-1)}$. Also note that n differs w.r.t. different pooling sizes.

Max pooling. A max pooling unit can be understood as returning the maximum response of a number of inputs, *i.e.* for two inputs A and B we have

$$C = \max(A, B). \tag{35}$$

For $A \sim \mathcal{N}(\mu_A, \sigma_A^2)$ and $B \sim \mathcal{N}(\mu_B, \sigma_B^2)$ the probability density of their maximum C according to [23]is given by

$$p(C = c) = \mathcal{N}(c \,|\, \mu_A, \sigma_A^2) \cdot \Phi\left(\frac{c - \mu_B}{\sigma_B}\right) + \mathcal{N}(c \,|\, \mu_B, \sigma_B^2) \cdot \Phi\left(\frac{c - \mu_A}{\sigma_A}\right), \tag{36a}$$

where

$$\Phi(y) = \int_{-\infty}^{y} \phi(x)\,\mathrm{d}x \qquad \text{and} \qquad \phi(y) = \frac{1}{\sqrt{2\pi}}\exp(-y^2/2) \tag{36b}$$

are the CDF and PDF of a unit-normal distribution.

Despite C not being normally distributed anymore, we approximate it by a univariate normal. In [23] the first and second moment are derived analytically. The mean is given by

$$\mu_C = \sqrt{\sigma_A^2 + \sigma_B^2} \cdot \phi(\alpha) + (\mu_A - \mu_B) \cdot \Phi(\alpha) + \mu_B \qquad \text{with} \qquad \alpha = \frac{\mu_A - \mu_B}{\sqrt{\sigma_A^2 + \sigma_B^2}} \tag{37a}$$

and the variance can be written as

$$v_C = (\mu_A + \mu_B)\sqrt{\sigma_A^2 + \sigma_B^2} \cdot \phi(\alpha) + (\mu_A^2 + \sigma_A^2) \cdot \Phi(\alpha) + (\mu_B^2 + \sigma_B^2) \cdot \big(1 - \Phi(\alpha)\big) - \mu_C^2. \tag{37b}$$

If we have more than two inputs, we concatenate the operations. More precisely, we fold any max pooling operation first in horizontal and then in vertical direction.

Rectified linear unit (ReLU). *c.f.* main paper.

Leaky rectifier (LeakyReLU). *c.f.* main paper.

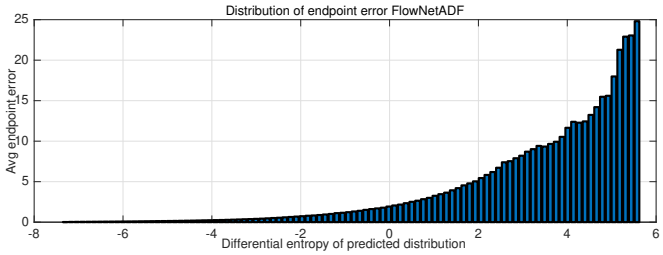
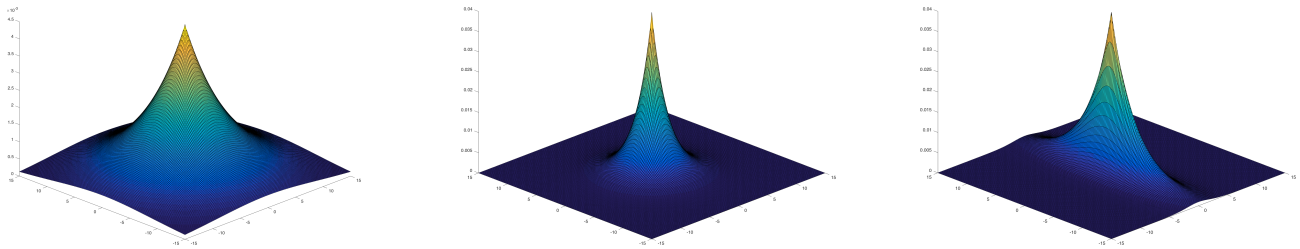


Figure 5. **Assessment of predicted regression uncertainties for an application in optical flow:** Differential entropy of the predictive distribution (x-axis) *vs*. average endpoint error (y-axis). A high entropy corresponds to a prediction with high variance. In this region the endpoint error is also large, suggesting that the uncertainty of the prediction is highly correlated to the actual error.

C. Calibration of Regression Uncertainties

Similar to the comparison of cross entropy *vs*. categorical entropy for classification in the main paper, we can assess the quality of the predictive distributions of our probabilistic optical flow networks in the case of regression. As already pointed out in the main paper, our predicted uncertainties are well correlated with the actual endpoint error, which suggests that our probabilistic approach is able to assess where it fails and where it succeeds. This is also borne out in Fig. 5, where we plot the endpoint error for FlowNetADF against the differential entropy for all images in the FlyingChairs test set. Hence, predicted variances can be reliably used to assess a model's accuracy.

D. Exponential Power Outputs



(a) Big (equal) variances (b) Small (equal) variances (c) Small (differing) variances

Figure 6. **Exponential power units for optical flow regression** in *FlowNetADF* and *FlowNetProbOut*. In this (artificial) illustrative example, the mean prediction is zero. (a) and (b) correspond to high and low variance predictions in which the variances of horizontal and vertical flow are equal, respectively. (c) shows the density for a case, where horizontal and vertical variances differ by an order of magnitude.

To gain some intuition on the workings of the probabilistic prediction, Fig. 6 shows a visualization of the probability density of the predictive distribution at an output node for the two-dimensional exponential power unit we used in the FlowNet networks of the main paper. The x-y axes correspond to u-v components of the optical flow. Fig. 6a shows the density of an uncertain prediction with large variance. Fig. 6b instead shows a (more) certain prediction with a smaller variance. In both cases, the variances for the horizontal and vertical flow directions are equal, respectively. Fig. 6c also corresponds to a smaller variance, however, here variances in horizontal and vertical direction differ by an order of magnitude. Note that unlike in a Gaussian output, the flow components in horizontal and vertical direction are not independent.

E. Additional Examples

Figs. 7 and 8 shows some examples from the CIFAR10 test set to illustrate how $\mathrm{softmax}$ predictions differ from predictions made with our lightweight probabilistic networks. Note that for our networks we extract the categorical distribution from the mean of the Dirichlet. While $\mathrm{softmax}$ predictions tend to be very confident in most cases, *i.e*. predictions have very low entropy with much of the probability mass concentrated on a single category, predictions made with the Dirichlet output layer rarely collapse to a single class in practice. Also, the predictions tend to have higher entropy, *i.e*. a more uniform distribution over classes, when the network makes an incorrect prediction. In other words our lightweight probabilistic networks have a sense of when they fail.

Figs. 11 and 12 give additional results for optical flow prediction on the Sintel dataset.

Figure 7. **Correctly classified CIFAR10 images** (test set). Blue bars indicate results from our ADF network with Dirichlet outputs (Dir + CLLH), while red bars indicate deterministic results with a softmax. Overall, the softmax results tend to be much more confident than the Dirichlet output layer, *i.e*. they often yield peaky predictions with very low entropy. While the Dirichlet output layer also assigns the correct Top-1 class, its predictions do not tend to be quite as confident as the softmax predictions, *i.e*. yielding higher entropy predictions.

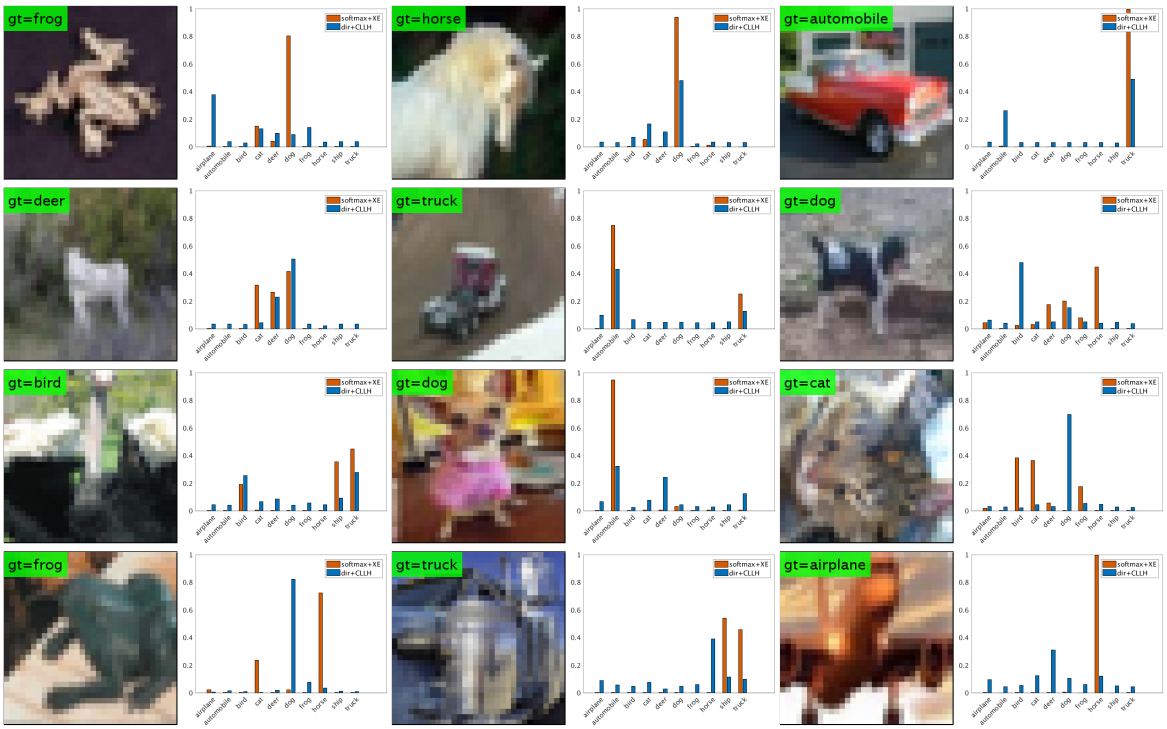


Figure 8. **Missclassified CIFAR10 images** (test set). Color coding as above. Although the predictions are incorrect, the deterministic softmax predictions are still highly confident (cf. frog top left, airplane bottom right). The Dirichlet layer also fails on these cases, yet its predictions are less confident and have higher entropy.

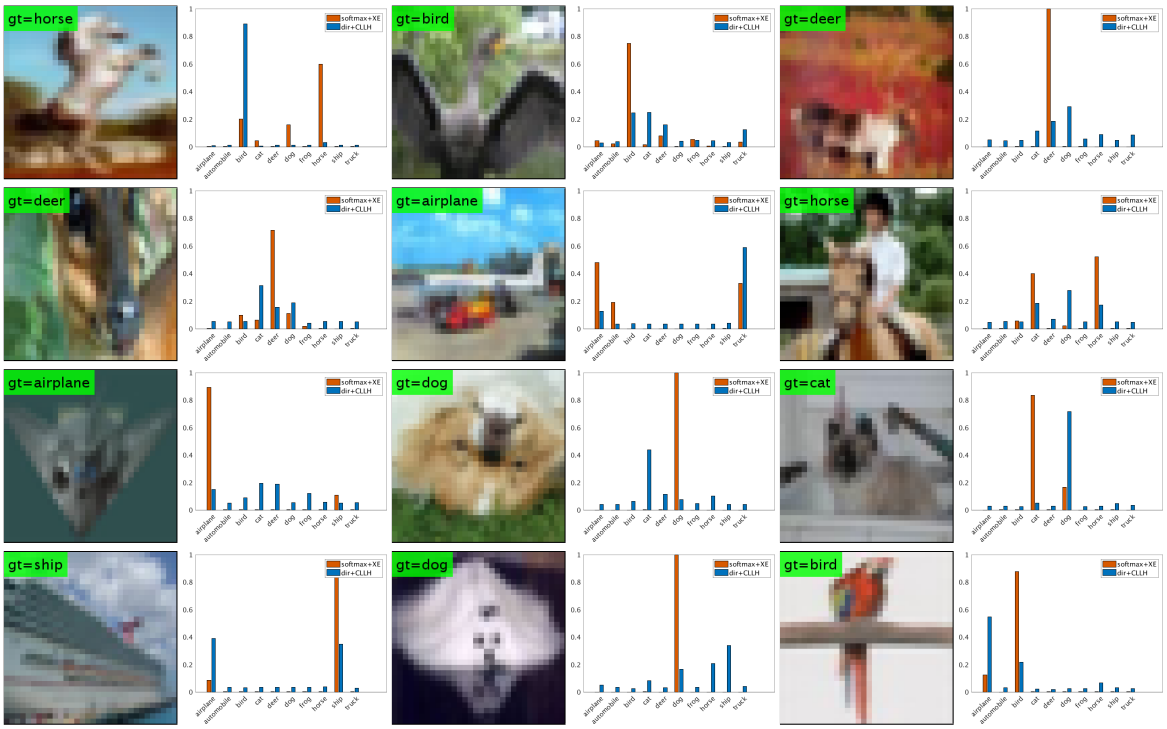


Figure 9. **Failure cases for Dirichlet layer**. Color coding as before. Here, the softmax predicts the correct classes, while the Dirichlet layer fails. However, in many cases the Dirichlet yields highly uncertain, *i.e*. high entropy, predictions.

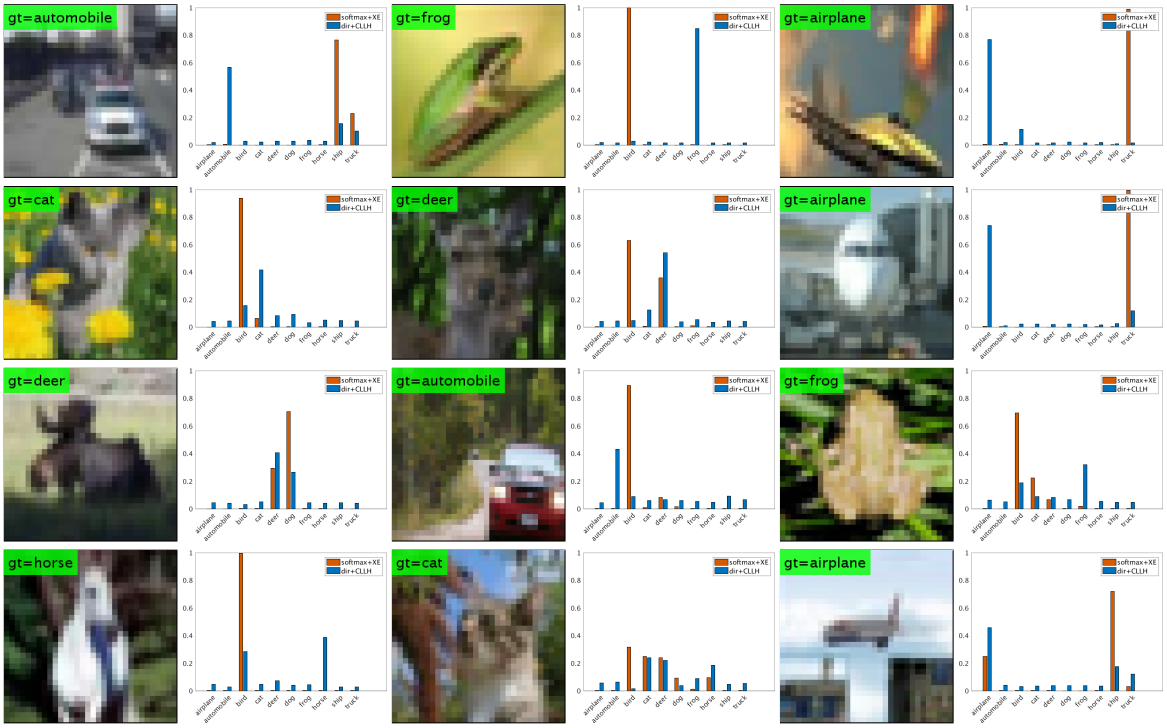
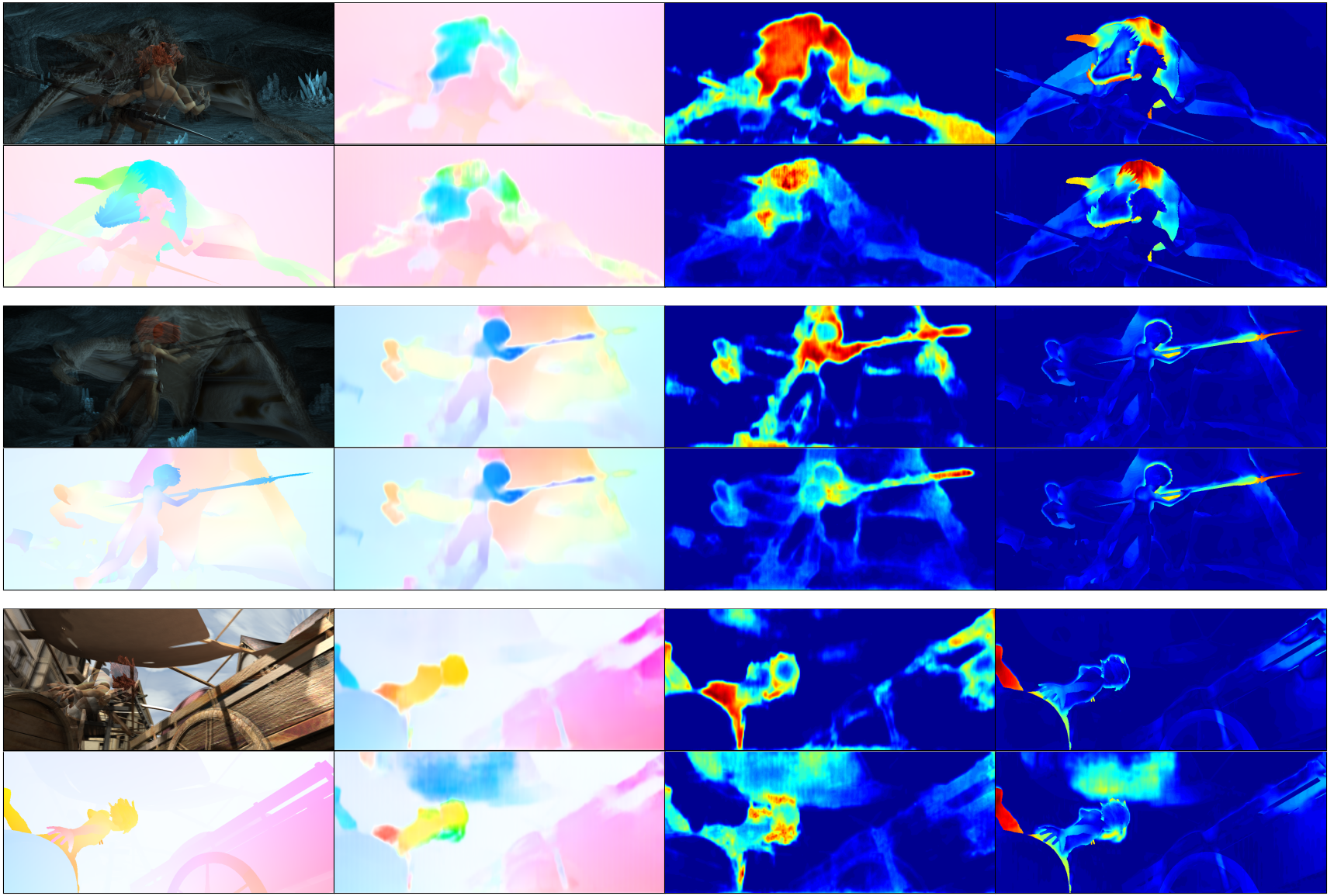
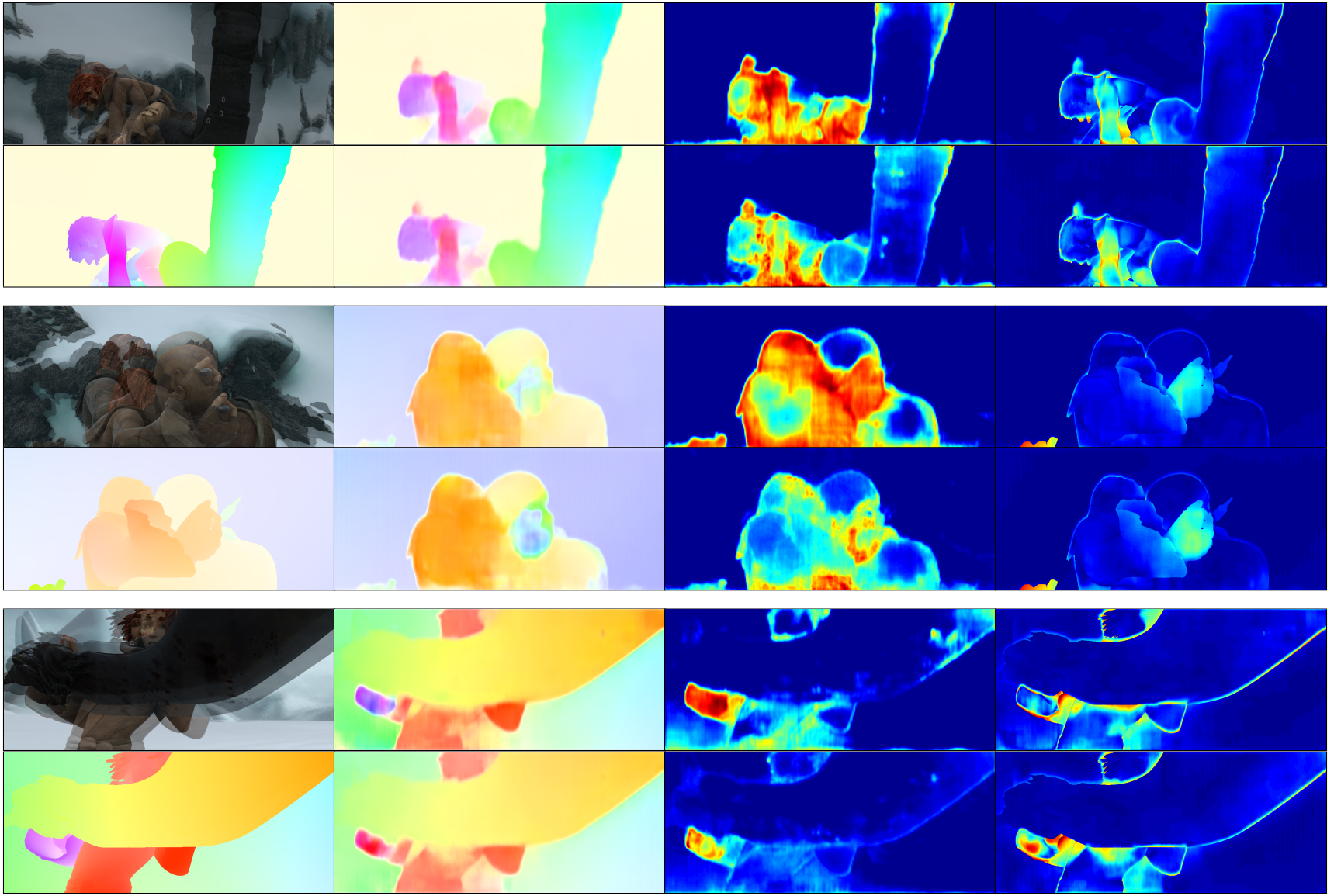


Figure 10. **Failure cases for softmax**. Color coding as before. Here, the softmax fails, while the Dirichlet layer succeeds. Note how confident softmax predictions tend to be, despite predicting the wrong class (*c.f*. airplanes top right).



(a) Inputs + GT (b) Predicted mean (c) Predicted uncertainty (d) Empirical EPE

Figure 11. **Probabilistic regression of optical flow on Sintel images.** Our lightweight probabilistic CNNs, FlowNetADF (top of each block) and FlowNetProbOut (bottom of each block), yield uncertainties for predictions while staying competitive w.r.t. the endpoint error (EPE). The uncertainties are visibly correlated with the EPE.



(a) Inputs + GT (b) Predicted mean (c) Predicted uncertainty (d) Empirical EPE

Figure 12. **Probabilistic regression of optical flow on Sintel images.** Our lightweight probabilistic CNNs, FlowNetADF (top of each block) and FlowNetProbOut (bottom of each block), yield uncertainties for predictions while staying competitive w.r.t. the endpoint error (EPE). The uncertainties are visibly correlated with the EPE.