

FastDraw: Addressing the Long Tail of Lane Detection by Adapting a Sequential Prediction Network

Jonah Philion

ISEE.AI

jonahphilon@isee.ai

Abstract

The search for predictive models that generalize to the long tail of sensor inputs is the central difficulty when developing data-driven models for autonomous vehicles. In this paper, we use lane detection to study modeling and training techniques that yield better performance on real world test drives. On the modeling side, we introduce a novel fully convolutional model of lane detection that learns to decode lane structures instead of delegating structure inference to post-processing. In contrast to previous works, our convolutional decoder is able to represent an arbitrary number of lanes per image, preserves the polyline representation of lanes without reducing lanes to polynomials, and draws lanes iteratively without requiring the computational and temporal complexity of recurrent neural networks. Because our model includes an estimate of the joint distribution of neighboring pixels belonging to the same lane, our formulation includes a natural and computationally cheap definition of uncertainty. On the training side, we demonstrate a simple yet effective approach to adapt the model to new environments using unsupervised style transfer. By training FastDraw to make predictions of lane structure that are invariant to low-level stylistic differences between images, we achieve strong performance at test time in weather and lighting conditions that deviate substantially from those of the annotated datasets that are publicly available. We quantitatively evaluate our approach on the CVPR 2017 Tusimple lane marking challenge, difficult CULane datasets [29], and a small labeled dataset of our own and achieve competitive accuracy while running at 90 FPS.

1. Introduction

Previous models of lane detection generally follow the following three-step template. First, the likelihood that each pixel is part of a lane is estimated. Second, pixels that clear a certain threshold probability p_{min} of being part of a lane are collected. Lastly, these pixels are clustered, for instance

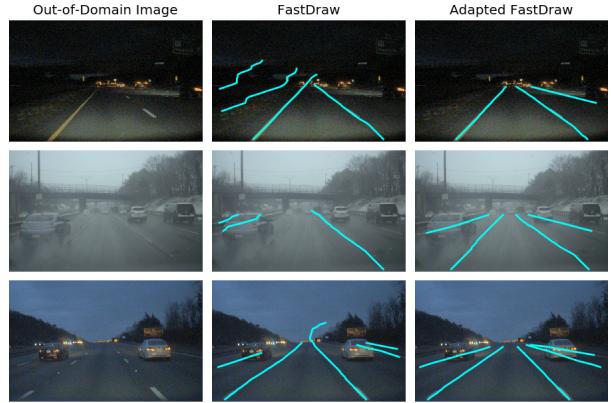


Figure 1. Best viewed in color. We train a novel convolutional lane detection network on a public dataset of labeled sunny California highways. Deploying the model in conditions far from the training set distribution (left) leads to poor performance (middle). Leveraging unsupervised style transfer to train FastDraw to be invariant to low-level texture differences leads to robust lane detection (right).

with RANSAC, into individual lanes.

Because the second and third steps in which road structure is inferred from a point cloud of candidate pixels are in general not differentiable, the performance of models of lane detection that follow this template is limited by the performance of the initial segmentation. We propose a new approach to lane detection in which the network performs the bulk of the decoding, thereby eliminating the need for hyper-parameters in post-processing. Our model “draws” lanes in the sense that the network is trained to predict the local lane shape at each pixel. At test time, we decode the global lane by following the local contours as predicted by the CNN.

A variety of applications benefit from robust lane detection algorithms that can perform in the wild. If the detector is iterative, the detector can be used as an interactive annotation tool which can be used to decrease the cost of building high definition maps [16, 1]. For level 5 systems that depend on high definition maps, online lane de-

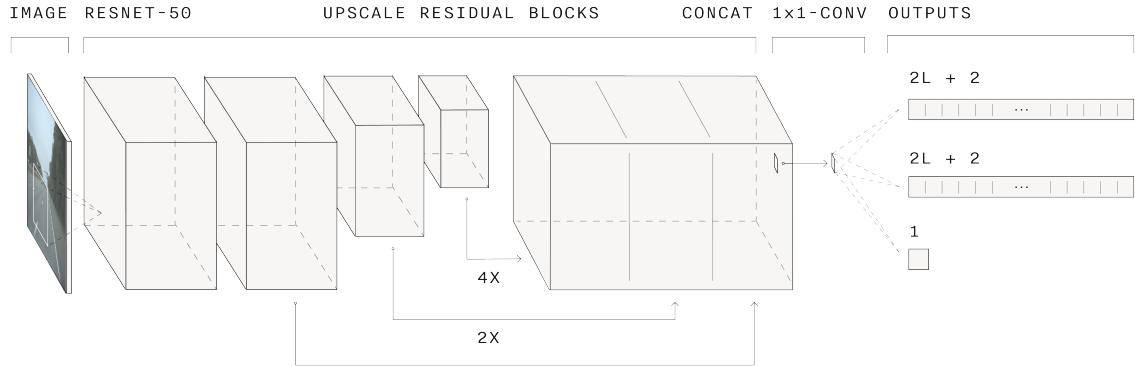


Figure 2. We use a CNN to extract a semantic representation of the input image. This representation is decoded by three separate shallow convolutional heads: a binary segmentation of pixels that belong to a lane ($p_{h,w,0}$), and a categorical distribution over the pixels within L pixels of the current pixel in the rows above and below ($p_{h,w,1}$ and $p_{h,w,-1}$ respectively). Because we include an end token in the categorical distribution to train to the network to predict endpoints, the categorical distributions are $2L + 1 + 1 = 2L + 2$ dimensional.

tection is a useful localization signal. Level 2 systems that are not equipped to handle the computational load required of high definition maps depend on models of lane detection equipped with principled methods of determining when to notify the driver that the lane detection is uncertain. In pursuit of solutions for these applications, we identify three characteristics that a lane detection module should possess.

First, the lane detection algorithm must be able to represent any number of lanes of any length. Whereas variability in the number of instances of an object in an image is an aspect of any kind detection problem, variability in the dimensionality of a single instance is a more unique to the lane detection problem; unlike bounding boxes which have a precise encoding of fixed dimensionality, lane segments can be arbitrary length. Solutions that reduce lanes to a constant dimensionality - such as by fitting them with polynomials - lose accuracy on tight curves where accurate lane detection or localization is important for safe driving.

Second, the detection algorithm must run in real-time. Therefore, although there is variability in the number and size of lanes in an image, whatever recursion used to identify and draw these lanes must be fast. Solutions to the variable dimensionality problem that involve recurrent cells [15] or attention [35] are therefore a last resort.

Finally, the detection algorithm must be able to adapt quickly to new scenes. Sensors such as cameras and lidar that are used in self-driving carry with them a long tail in the distribution of their outputs. A lane detection algorithm should be able to adapt to new domains in a scalable way.

We present an approach which addresses these problems and is competitive with other contemporary lane detection algorithms. Our contributions are

- A lane detection model that integrates the decoding step directly into the network. Our network is autore-

gressive and therefore comes equipped with a natural definition of uncertainty. Because decoding is largely carried out by the convolutional backbone, we are able to optimize the network to run at 90 frames per second on a GTX 1080. The convolutional nature of FastDraw makes it ideal for multi-task learning [20] or as an auxiliary loss [6].

- A simple but effective approach to adapt our model to handle images that are far from the distribution of images for which we have public annotations. Qualitative results are shown in Figure 1 and Figure 6. While style transfer has been used extensively to adapt the output distribution of simulators to better match reality [10], we use style transfer to adapt the distribution of images from publicly available annotated datasets to better match corner case weather and environmental conditions.

2. Related Work

Lane Detection Models of lane detection generally involve extracting lane marking features from input images followed by clustering for post-processing. On well-maintained roads, algorithms using hand-crafted features work well [2, 19]. Recent approaches such as those that achieve top scores on the 2017 Tusimple Lane Detection Challenge seek to learn these hand-crafted features in a more end-to-end manner using convolutional neural networks. To avoid clustering, treating left-left lane, left lane, right lane, and right-right lane as channels of the segmentation has been explored [29]. Projecting pixels onto a ground plane via a learned homography is a strong approach for regularizing the curve fitting of individual lanes [28]. Research into improving the initial segmentation has been suc-

cessful although results are sensitive to the heuristics used during post-processing [12]. Recent work has improved segmentation by incorporating a history of images instead of exclusively conditioning on the current frame [39].

Lane detection is not isolated to dashcam imagery. Models that detect lanes in dashcams can in general be adapted to detect lanes in lidar point-clouds, open street maps, and satellite imagery [16, 23, 4]. The success of semantic segmentation based approaches to lane detection has benefited tremendously from rapid growth in architectures that empirically perform well on dense segmentation tasks [5, 26, 25]. Implicitly, some end-to-end driving systems have been shown to develop a representation of lanes without the need for annotated data [21, 7].

Style Transfer Adversarial loss has enabled rapid improvements in a wide range of supervised and unsupervised tasks. Pix2Pix [18] was the first to demonstrate success in style translation tasks on images. Unsupervised style transfer for images [27, 38] and unsupervised machine translation [24, 3] use back-translation as a proxy for supervision. While models of image-to-image translation have largely been deterministic [27, 38], MUNIT [17] extends these models to generate a distribution of possible image translations. In this work, we incorporate images generated by MUNIT translated from a public dataset to match the style of our own cameras. We choose MUNIT for this purpose because it is unsupervised and generative. Previous work has used data from GTA-V to train object detectors that operate on lidar point clouds [37]. Parallel work has shown that synthetic images stylized by MUNIT can improve object detection and semantic segmentation [10]. We seek to leverage human annotated datasets instead of simulators as seeds for generating pseudo training examples of difficult environmental conditions.

Drawing We take inspiration from work in other domains where targets are less structured than bounding boxes such as human pose estimation [8] and automated object annotation [1]. In human pose estimation, the problem of clustering joints into human poses has been solved by inferring slope fields between body parts that belong to the same human [8]. Similarly, we construct a decoder which predicts which pixels are part of the same lane in addition to a segmentation of lanes. In Polygon-RNN [9] and Sketch-RNN [13], outlines of objects are inferred by iteratively drawing bounding polygons. We follow a similar model of learned decoding while simplifying the recurrence due to the relative simplicity of the lane detection task and need for real-time performance.

3. Model

Our model maximizes the likelihood of polylines instead of purely predicting per-pixel likelihoods. In doing so, we avoid the need for heuristic-based clustering post-

processing steps [2]. In this section, we describe how we derive our loss, how we decode lanes from the model, and how we train the network to be robust to its own errors when conditioning on its own predictions at test time.

3.1. Lane Representation

In the most general case, lane annotations are curves $\gamma : [0, 1] \rightarrow \mathbb{R}^2$. In order to control the orientation of the lanes, we assume that lane annotations can be written as a function of the vertical axis of the image. A lane annotation \mathbf{y} therefore is represented by a sequence of $\{\text{height}, \text{width}\}$ pixel coordinates $\mathbf{y} = \{y_1, \dots, y_n\} = \{\{h_1, w_1\}, \dots, \{h_n, w_n\}\}$ where $h_{i+1} - h_i = 1$.

Given an image $\mathbf{x} \in \mathbb{R}^{3 \times H \times W}$, the joint probability $p(\mathbf{y}|\mathbf{x})$ can be factored

$$p(\mathbf{y}|\mathbf{x}) = p(y_1|\mathbf{x}) \prod_{i=1}^{n-1} p(y_{i+1}|y_1, \dots, y_i, \mathbf{x}). \quad (1)$$

One choice to predict $p(y_{i+1}|y_1, \dots, y_i, \mathbf{x})$ would be to use a recurrent neural network [16, 36]. To decode quickly, we assume most of the dependency can be captured by conditioning only on the previous decoded coordinate

$$p(\mathbf{y}|\mathbf{x}) \approx p(y_1|\mathbf{x}) \prod_{i=1}^{n-1} p(y_{i+1}|y_i, \mathbf{x}). \quad (2)$$

Because we assume $h_{i+1} - h_i = 1$, we can simplify

$$p(y_{i+1}|y_i, \mathbf{x}) = p(\Delta w_i|y_i, \mathbf{x}) \quad (3)$$

$$\Delta w_i = w_{i+1} - w_i. \quad (4)$$

Lane detection is then reduced to predicting a distribution over dw/dh at every pixel in addition to the standard per-pixel likelihood. Decoding proceeds by choosing an initial pixel coordinate and integrating.

To represent the distribution $p(\Delta w_i|y_i, \mathbf{x})$, we could use a normal distribution and perform regression. However, in cases where the true distribution is multi-modal such as when lanes split, a regression output would cause the network to take the mean of the two paths instead of capturing the multimodality. Inspired by WaveNet [34], we choose to make no assumptions about the shape of $p(\Delta w_i|y_i, \mathbf{x})$ and represent the pairwise distributions using categorical distributions with support $\Delta w \in \{i \in \mathbb{Z} | -L \leq i \leq L\} \cup \{\text{end}\}$ where L is chosen large enough to be able to cover nearly-horizontal lanes and `end` is a stop token signaling the end of the lane. At each pixel $\{h, w\}$, our network predicts

- $p_{h,w,0} := p(y_{h,w} = 1|\mathbf{x})$ - the probability that pixel $\{h, w\}$ is part of a lane.
- $p_{h,w,1} := p(\Delta w^{+1}|y_{h,w} = 1, \mathbf{x})$ - the categorical distribution over pixels in the row **above** pixel $\{h, w\}$

within a distance L that pixel $\{h + 1, w + \Delta w^{+1}\}$ is part of the same lane as pixel $\{h, w\}$ or that pixel $\{h, w\}$ is the **top** pixel in the lane it is a part of.

- $p_{h,w,-1} := p(\Delta w^{-1}|y_{h,w} = 1, \mathbf{x})$ - the categorical distribution over pixels in the row **below** pixel $\{h, w\}$ within a distance L that pixel $\{h-1, w+\Delta w^{-1}\}$ is part of the same lane as pixel $\{h, w\}$ or that pixel $\{h, w\}$ is the **bottom** pixel in the lane it is a part of.

Given these distributions, we can quickly decode a full lane segment given any initial point on the lane. Given some initial position h_0, w_0 on lane \mathbf{y} , we follow the greedy recursion

$$\mathbf{y}(h_0) = w_0 \quad (5)$$

$$\mathbf{y}(h+d) = \mathbf{y}(h) + \Delta w^d \quad (6)$$

$$\Delta w^d = -L + \text{argmax}_{h,y(h),d} \quad (7)$$

where $d \in \{-1, 1\}$ during the downwards and upwards drawing stages. Note that we can choose any $y_i \in \mathbf{y}$ as h_0, w_0 as long as we concatenate the results from the upwards and downwards trajectories. We stop decoding when argmax returns the `end` token.

3.2. Architecture

To extract a semantic representation of the input image, we repurpose Resnet50/18 [14] for semantic segmentation. The architecture is shown in Figure 2. We use two skip connections to upscale and concatenate features at a variety of scales. All three network heads are parameterized by two layer CNNs with kernel size 3. In all experiments, we initialize with Resnets pretrained on Imagenet [30].

3.3. Loss

We minimize the negative log likelihood given by (2). Let θ represent the weights of our network, $\mathbf{x} \in \mathbb{R}^{3,H,W}$ an input image, $\mathbf{y} = \{\{h_1, w_1\}, \dots, \{h_n, w_n\}\}$ a ground truth lane annotation such that $h_i - h_{i-1} = 1$ and $\mathbf{y}_m \in \mathbb{R}^{1,H,W}$ a ground truth segmentation mask of the lane. The loss $L(\theta)$ is defined by

$$L_{mask}(\theta) = -\log(p(\mathbf{y}_m | f_\theta(\mathbf{x}))) \quad (8)$$

$$L_{sequence}(\theta) = -\sum_{d \in \{-1, 1\}} \sum_{i=1}^n \log(p(\Delta w_i^d | f_\theta(\mathbf{x}))) \quad (9)$$

$$L(\theta) = L_{mask}(\theta) + L_{sequence}(\theta) \quad (10)$$

Because the task of binary segmentation and pairwise prediction have different uncertainties and scales, we dynamically weight these two objectives [20]. We incorporate

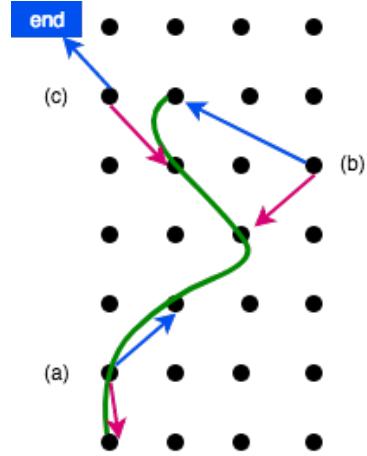


Figure 3. In addition to predicting per-pixel likelihoods, we train our network to output the pixels in the row above (blue) and below (purple) that are in the same lane as the current pixel. We also train pixels that are offset from annotated lanes to point back to the annotated lane (b). We include the `end` token in the categorical distribution to signal the termination of a lane (c). Given these predictions, we draw lanes by sampling an initial point then greedily following arrows up and down until we reach `end` in either direction and concatenating the two results.

a learned temperature σ which is task specific to weigh our loss:

$$L(\theta) = \frac{1}{\sigma_{mask}^2} L_{mask}(\theta) + \frac{1}{\sigma_{sequence}^2} L_{sequence}(\theta) + \log \sigma_{mask}^2 \sigma_{sequence}^2. \quad (11)$$

During training, we substitute $W = \log \sigma^2$ into (11) for numerical stability. Experiments in which we fixed W resulted in similar performance to allowing W to be learnable. However, we maintain the dynamically weighed loss for all results reported below to avoid tuning hyperparameters.

3.4. Exposure Bias

Because our model is autoregressive, training $p_{h,w,\pm 1}$ by conditioning exclusively on ground truth annotations leads to drifting at test time [32]. One way to combat this issue is to allow the network to act on its own proposed decoding and use reinforcement learning to train. The authors of [1] take this approach using self-critical sequence training [33] and achieve good results.

Although we experiment with reinforcement learning, we find that training the network to denoise lane annotations — also known as augmenting datasets with “synthesized perturbations” [6] — is significantly more sample efficient. This technique is visualized in Figure 3. To each ground truth annotation \mathbf{y} we add gaussian noise and train the network to predict the same target as the pixels in \mathbf{y} . We

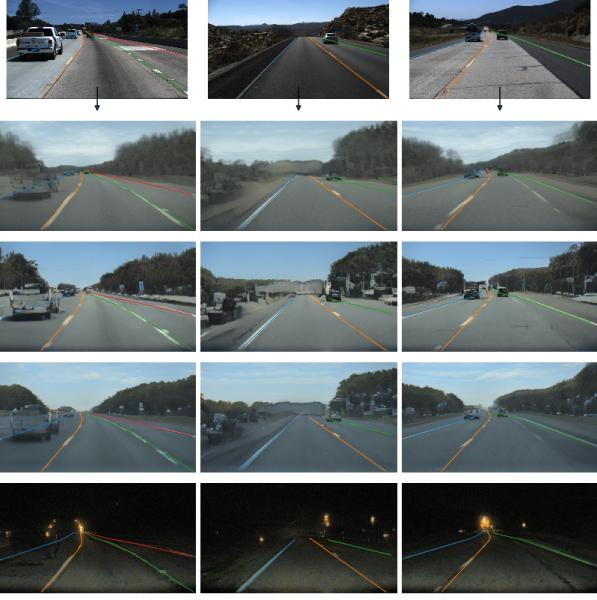


Figure 4. The top row shows three images \mathbf{x}_i from the Tusimple dataset and their annotations. The bottom four rows display samples from $G(\mathbf{x}_i)$ with the adjusted Tusimple annotations overlaid. We use these additional training samples to bias the network towards shape instead of texture [11].

therefore generate training examples

$$s \sim [\mathcal{N}(0.5, \sigma)] \quad (12)$$

$$w'_i = w_i + s \quad (13)$$

$$\Delta w_i^d = w_{i+s} - w'_i \quad (14)$$

where $d \in \{-1, 1\}$. We tune σ as a hyperparameter which is dependent on dataset and image size. We clamp all Δw between $-L$ and L and clamp all w' between 0 and the width of the image.

3.5. Adaptation

The downside of data-driven approaches is that we have weak guarantees on performance once we evaluate the model on images far from the distribution of images that it trained on. For this purpose, we leverage the MUNIT framework [17] to translate images from public datasets with ground truth annotations into a distribution of images we acquire by driving through Massachusetts in a variety of weather and lighting conditions.

To perform style transfer on the images in unordered datasets D and D' , the CycleGAN framework [38] trains an encoder-generator pair E, G for each dataset D and D' such that $G(E(\mathbf{x})) \approx \mathbf{x}$ for $\mathbf{x} \sim D$ and difference between the distributions $y \sim G'(E(\mathbf{x}))$ and $y \sim D'$ is minimized, with analogous statements for D' . The MUNIT framework generalizes this model to include a style vector $s \sim \mathcal{N}(0, \mathbf{I})$

as input to the encoder E . Style translations are therefore distributions that can be sampled from instead of deterministic predictions.

As shown in Figure 4, we use MUNIT to augment our labeled training set with difficult training examples. Let $D = \{\mathbf{x}_i, \mathbf{y}_i\}$ be a dataset of images \mathbf{x}_i and lane annotations \mathbf{y}_i and $D' = \{\mathbf{x}_i\}$ a corpus of images without labels. Empirically, we find that style transfer preserves the geometric content of input images. We can therefore generate new training examples $\{\mathbf{x}', \mathbf{y}'\}$ by sampling from the distribution $D' \sim \{\mathbf{x}', \mathbf{y}'\}$ defined by

$$\mathbf{x}, \mathbf{y} \sim D \quad (15)$$

$$\mathbf{x}' \sim G'(\mathbf{x}, s)_{s \sim \mathcal{N}(0, \mathbf{I})} \quad (16)$$

$$\mathbf{y}' = \mathbf{y} \quad (17)$$

Although representation of lanes around the world are location dependent, we theorize that the distribution of lane geometries is constant. Unsupervised style transfer allows us to adjust to different styles and weather conditions without the need for additional human annotation.

4. Experiments

We evaluate our lane detection model on the Tusimple Lane Marking Challenge and the CULane datasets [29]. The Tusimple dataset consists of 3626 annotated 1280x720 images taken from a dash camera as a car drives on California highways. The weather is exclusively overcast or sunny. We use the same training and validation split as ELLGAN [12]. In the absence of a working public leaderboard, we report results exclusively on the validation set. We use the publicly available evaluation script to compute accuracy, false positive rate, and true positive rate.

Second, we adopt the same hyperparameters determined while training on Tusimple and train our network on the challenging CULane dataset. CULane consists of 88880 training images, 9675 validation images, and 34680 test images in a diversity of weather conditions scenes. The test set includes environmental metadata about images such as if the image is crowded, does not have lane lines, or is tightly curved. We report evaluation metrics on each of these cases as is done by CULane [29].

Finally, to evaluate the effectiveness of our model to adapt to new scenes, we drive in Massachusetts in a diversity of weather conditions and record 10000 images of dash cam data. Using publicly available source code, we train MUNIT to translate between footage from the Tusimple training set and our own imagery, then sample 10000 images from the generator. We note that upon evaluating qualitatively the extent to which the original annotations match the generated images, the frame of the camera is transformed. We therefore scale and bias the height coordinates of the original annotations with a single scale and



Figure 5. The standard deviation of the distribution predicted by FastDraw is plotted as error bars on a variety of images from Tusimple test set. Our color map is thresholded at a standard deviation of 0 and 9 pixels. We find that the network is accurately growing more uncertain in regions where the exact location of a lane is not well defined, for instance when the lane marking is wide, there are shadows, the lane is possibly a false positive, or the lane is occluded by other vehicles.

bias across all images to develop D' .

$$w_i^* = mw_i + b \quad (18)$$

Samples from the pseudo training example generator are shown in Figure 4. A speed comparison of FastDraw to other published models is shown in Table 2. For models trained on images of size 128×256 we use $L = 6$ pixels and $\sigma = 2$ pixels. For models trained on images of size 352×640 we use $L = 16$ pixels and $\sigma = 5$ pixels.

5. Results

5.1. Tusimple

We train FastDraw on the Tusimple dataset. We train for 7 epochs in all experiments with batch size 4 using the Adam optimizer [22]. We initialize learning rate to 1.0e-4 and halve the learning rate every other epoch. To generate the candidates for initial points of lanes, we mask the center of the image and use DBSCAN from scikit-learn [31] with $\epsilon = 5$ pixels to cluster. Given these initial positions, we follow the predictions of FastDraw to draw the lane upwards and downwards.

We compare our algorithm quantitatively against EL-GAN [12]. EL-GAN improves lane detection over traditional binary segmentation by adding an additional adversarial loss to the output segmentation. The segmentations therefore approximate the distribution of possible labels. However, this approach still requires a heuristic decoder to convert the segmentation into structured lane objects. FastDraw is competitive with EL-GAN by all Tusimple metrics as shown in Table 1.

| Method | Acc (%) | FP | FN |
|-----------------------------|-------------|--------------|--------------|
| EL-GAN (basic) | 93.3 | 0.061 | 0.104 |
| EL-GAN (basic++) | 94.9 | 0.059 | 0.067 |
| FastDraw Resnet18 | 94.9 | 0.061 | 0.047 |
| FastDraw Resnet50 | 94.9 | 0.059 | 0.052 |
| FastDraw Resnet50 (adapted) | 95.2 | 0.076 | 0.045 |

Table 1. We use Tusimple evaluation metrics to compare quantitatively with EL-GAN [12]. FastDraw achieves comparable performance to EL-GAN with fewer layers. We note that while the accuracy of adapted FastDraw achieves high accuracy, it also has the highest false positive rate. We reason that the network learns a stronger prior over lane shape from D' , but the style segmentation does not always preserve the full number of lanes which results in the side of the road being falsely labeled a lane in the D' dataset.

5.2. Uncertainty

Because our network predicts a categorical distribution at each pixel with support $\Delta w \in \{i \in \mathbb{Z} | -L \leq i \leq L\} \cup \{\text{end}\}$, we can quickly compute the standard deviation of this distribution conditioned on $\Delta w \neq \text{end}$ and interpret the result as errorbars in the width dimension. We find that the uncertainty in the network increases in occluded and shadowy conditions. Example images are shown in Figure 5. These estimates can be propagated through the self-driving stack to prevent reckless driving in high-uncertainty situations.

5.3. Is the learned decoder different from a simple heuristic decoder?

A simple way to decode binary lane segmentations is to start at an initial pixel, then choose the pixel in the row above the initial pixel with the highest probability of belonging to a lane. To show that our network is not following this decoding process, we calculate the frequency at which the network chooses pixels that agree with this simple heuristic based approach. The results are shown in Table 3. We find that although the output of the two decoders are correlated, the learned decoder is in general distinct from the heuristic decoder.

| Model | Frames-Per-Second |
|------------------|-------------------|
| Ours | 90.31 |
| H-Net [20] | 52.6 |
| CULane [21] | 17.5 |
| PolyLine-RNN [9] | 5.7 |
| EL-GAN [7] | <10 |

Table 2. Because FastDraw requires little to no post-processing, the runtime is dominated by the forward pass of the CNN backbone which can be heavily optimized.

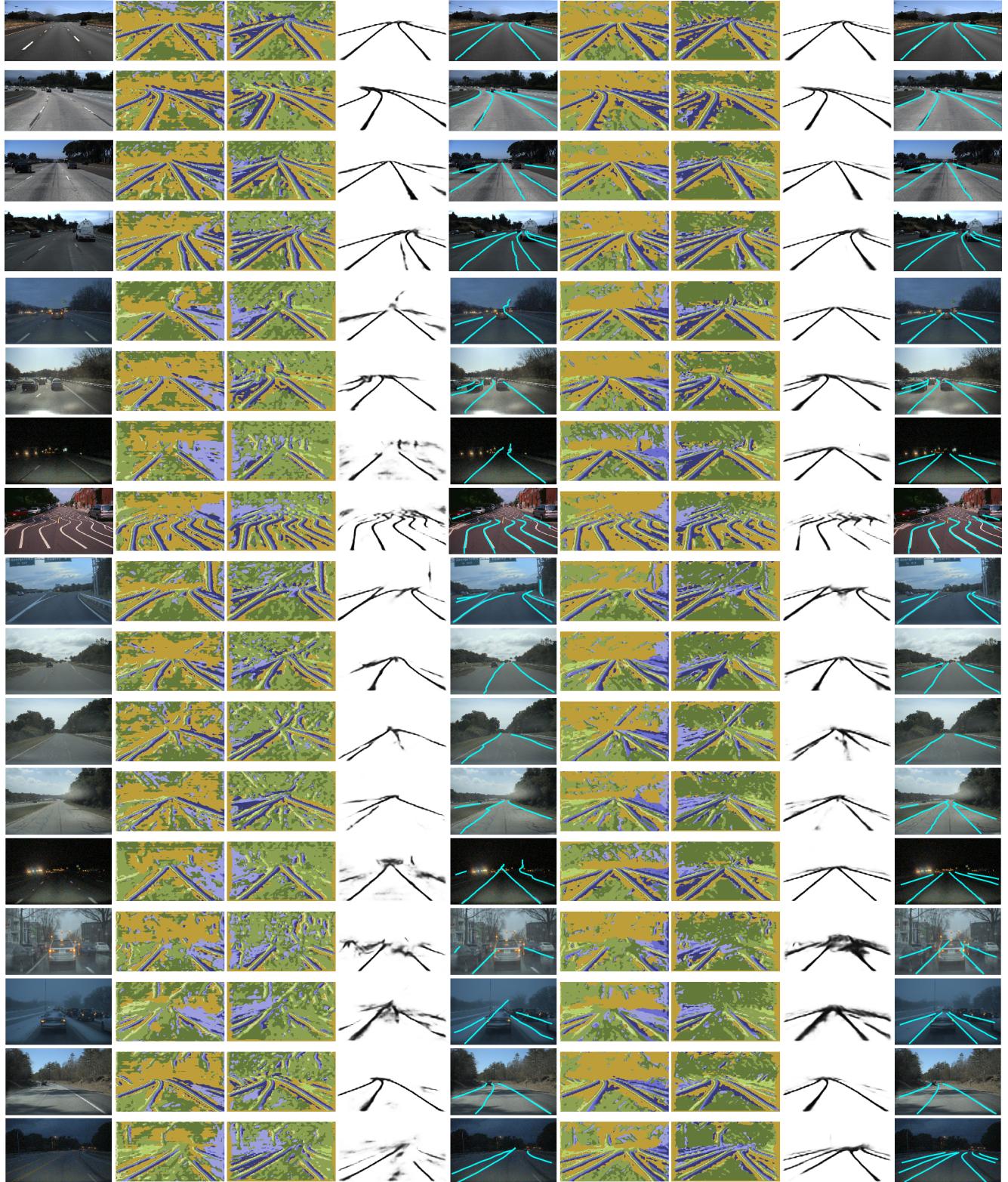


Figure 6. We demonstrate that our network can perform with high accuracy on the publicly annotated dataset we train on as well as images collected from a diversity of weather conditions and scenes in Massachusetts. Column 1 shows the “lookup” head $\text{argmax}_{h,w,1}$, column 2 shows the “lookdown” head $\text{argmax}_{h,w,-1}$, column 3 shows the per-pixel likelihood $p_{h,w,0}$, and column 4 shows the decoded lanes on top of the original image. Columns 4-8 show the analogous visual for adapted FastDraw. Images in the top 4 rows come from the Tusimple test set. The rest come from our own driving data collected in the wild on Massachusetts highways. We find that the data augmentation trains the network to draw smoother curves, recognize the ends of lanes better, and find all lanes in the image. Even on Tusimple images, the annotation improves the per-pixel likelihood prediction. At splits, the network chooses chooses a mode of the distribution instead of taking the mean.

| | $ p_{h,w,1} - \text{argmax} p_{h+1,w+\Delta w,0} $ | % |
|-----|--|------|
| < 1 | | 12.6 |
| < 3 | | 58.2 |
| < 5 | | 87.1 |

Table 3. For a series of decoded polylines, we calculate the distance between the pixel with maximum likelihood $p_{h\pm 1,w,0}$ in the row above $\{h, w\}$ with the chosen $\text{argmax} p_{h,w,\pm 1}$. We report the percent of the time that the distance is less than one, three, and five pixels. We find that the network is in general predicting values which agree with the $p_{h,w,0}$ predictions. Deviation from the naive decoder explains why the network still performs well when the segmentation mask is noisy.

| | ResNet-50 [29] | FastDraw Resnet50 |
|-----------|----------------|-------------------|
| Normal | 87.4 | 85.9 |
| Crowded | 64.1 | 63.6 |
| Night | 60.6 | 57.8 |
| No line | 38.1 | 40.6 |
| Shadow | 60.7 | 59.9 |
| Arrow | 79.0 | 79.4 |
| Dazzle | 54.1 | 57.0 |
| Curve | 59.8 | 65.2 |
| Crossroad | 2505 | 7013 |

Table 4. We compare FastDraw trained on CULane dataset to Resnet-50 on the CULane test set. We do not filter the lane predictions from FastDraw and achieve competitive results. While these scores are lower than those of SCNN [29], we emphasize that architectural improvements such as those introduced in [29] are complementary to the performance of the FastDraw decoder.

5.4. CULane

We train FastDraw on the full CULane training dataset [29]. We use identical hyperparameters to those determined on Tusimple with an exponential learning rate schedule with period 1000 gradient steps and multiplier 0.95. FastDraw finds the model achieves competitive performance with the CULane Resnet-50 baseline. IoU at an overlap of 0.5 as calculated by the open source evaluation provided by CULane is shown in Table 4. Of note is that FastDraw outperforms the baseline on curves by a wide margin as expected given that CULane assumes lanes will be well represented by cubic polynomials while FastDraw preserves the polyline representation of lanes.

5.5. Massachusetts

We evaluate the ability of FastDraw to generalize to new scenes. In Figure 6 we demonstrate qualitatively that the network trained on style transferred training examples in addition to the Tusimple training examples can generalize well to night scenes, evening scenes, and rainy scenes. We emphasize that no additional human annotation was

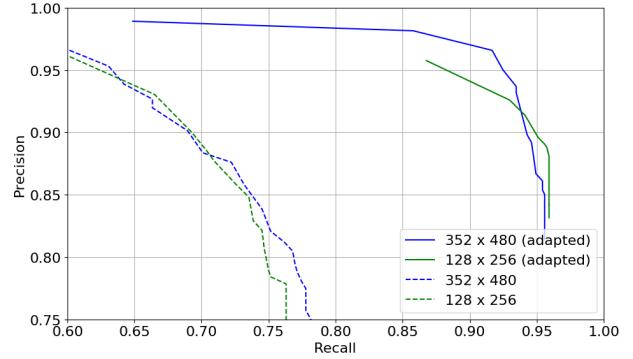


Figure 7. We label lane boundaries in a 300 image dataset of “long tail” images to quantitatively evaluate the effect of the data augmentation. The precision/recall tradeoff for models trained on the augmented dataset is much better than the tradeoff of models trained without. The effect of the adaptation is demonstrated qualitatively in Figure 6.

required to train FastDraw to be robust to these difficult environments.

Additionally, we plot the precision/recall trade-off of FastDraw models trained with and without adaptation in Figure 7. We use the same definition of false positive and false negative as used in the Tusimple evaluation. The augmentation compiles models that are markedly more robust to scene changes. We believe that these results echo recent findings that networks trained to do simple tasks on simple datasets learn low-level discriminative features that do not generalize [11]. Unsupervised style transfer for data augmentation is offered as a naive but effective regularization of this phenomenon.

6. Conclusion

We demonstrate that it is possible to build an accurate model of lane detection that can adapt to difficult environments without requiring additional human annotation. The primary assumption of our model is that lanes are curve segments that are functions of the height axis of an image and that a lane can be drawn iteratively by conditioning exclusively on the previous pixel that was determined to be part of the lane. With these assumptions, we achieve high accuracies on the lane detection task in standard and difficult environmental conditions.

References

- [1] D. Acuna, H. Ling, A. Kar, and S. Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. *CoRR*, abs/1803.09693, 2018. [1](#), [3](#), [4](#)
- [2] M. Aly. Real time detection of lane markers in urban streets. *CoRR*, abs/1411.7113, 2014. [2](#), [3](#)

- [3] M. Artetxe, G. Labaka, E. Agirre, and K. Cho. Unsupervised neural machine translation. *CoRR*, abs/1710.11041, 2017. 3
- [4] S. M. Azimi, P. Fischer, M. Körner, and P. Reinartz. Aerial lanenet: Lane marking semantic segmentation in aerial imagery using wavelet-enhanced cost-sensitive symmetric fully convolutional neural networks. *CoRR*, abs/1803.06904, 2018. 3
- [5] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015. 3
- [6] M. Bansal, A. Krizhevsky, and A. S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *CoRR*, abs/1812.03079, 2018. 2, 4
- [7] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016. 3
- [8] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. 3
- [9] L. Castrejón, K. Kundu, R. Urtasun, and S. Fidler. Annotating object instances with a polygon-rnn. *CoRR*, abs/1704.05548, 2017. 3
- [10] A. Dundar, M. Liu, T. Wang, J. Zedlewski, and J. Kautz. Domain stylization: A strong, simple baseline for synthetic to real image domain adaptation. *CoRR*, abs/1807.09384, 2018. 2, 3
- [11] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *CoRR*, abs/1811.12231, 2018. 5, 8
- [12] M. Ghafoorian, C. Nugteren, N. Baka, O. Booij, and M. Hofmann. El-gan: Embedding loss driven generative adversarial networks for lane detection, 2018. 3, 5, 6
- [13] D. Ha and D. Eck. A neural representation of sketch drawings. *CoRR*, abs/1704.03477, 2017. 3
- [14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 4
- [15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997. 2
- [16] N. Homayounfar, W.-C. Ma, S. Kowshika Lakshmikanth, and R. Urtasun. Hierarchical recurrent attention networks for structured online maps. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 3
- [17] X. Huang, M. Liu, S. J. Belongie, and J. Kautz. Multimodal unsupervised image-to-image translation. *CoRR*, abs/1804.04732, 2018. 3, 5
- [18] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016. 3
- [19] S. Jung, J. Youn, and S. Sull. Efficient lane detection based on spatiotemporal images. *IEEE Transactions on Intelligent Transportation Systems*, 17(1):289–295, Jan 2016. 2
- [20] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *CoRR*, abs/1705.07115, 2017. 2, 4
- [21] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J. Allen, V. Lam, A. Bewley, and A. Shah. Learning to drive in a day. *CoRR*, abs/1807.00412, 2018. 3
- [22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 6
- [23] A. Laddha, M. K. Kocamaz, L. E. Navarro-Sermen, and M. Hebert. Map-supervised road detection. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 118–123, June 2016. 3
- [24] G. Lample, L. Denoyer, and M. Ranzato. Unsupervised machine translation using monolingual corpora only. *CoRR*, abs/1711.00043, 2017. 3
- [25] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. 3
- [26] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. 3
- [27] M. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. *CoRR*, abs/1703.00848, 2017. 3
- [28] D. Neven, B. D. Brabandere, S. Georgoulis, M. Proesmans, and L. V. Gool. Towards end-to-end lane detection: an instance segmentation approach. *CoRR*, abs/1802.05591, 2018. 2
- [29] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang. Spatial as deep: Spatial CNN for traffic scene understanding. *CoRR*, abs/1712.06080, 2017. 1, 2, 5, 8
- [30] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 4
- [31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 6
- [32] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732, 2016. 4
- [33] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. Self-critical sequence training for image captioning. *CoRR*, abs/1612.00563, 2016. 4
- [34] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016. 3
- [35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. 2
- [36] Z. Wang, W. Ren, and Q. Qiu. Lanenet: Real-time lane detection networks for autonomous driving, 2018. 3
- [37] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer. Squeeze-segv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud, 2018. 3

- [38] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017. [3](#), [5](#)
- [39] Q. Zou, H. Jiang, Q. Dai, Y. Yue, L. Chen, and Q. Wang. Robust lane detection from continuous driving scenes using deep neural networks. *CoRR*, abs/1903.02193, 2019. [3](#)