FastDraw: Lane Detection by a Sequential Prediction Network

Jonah Philion
iSee.ai
jonahphilion@isee.ai

Wongun Choi
iSee.ai
wchoi@isee.ai

Yibiao Zhao
iSee.ai
yz@isee.ai

Abstract

Fully differentiable modules are integral to the perception stacks of modern self driving car teams. Although increasing the extent to which a pipeline is learned from data can increase accuracy, data-driven approaches have weak guarantees on performance when evaluated in new conditions. In this paper, we introduce an fully convolutional model of lane detection and demonstrate a simple yet effective approach to adapt the model to generalize to new environments. In contrast to previous works, our convolutional decoder is able to represent an arbitrary number of lanes per image, preserves the waypoint 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. Using only publicly available human annotations of lane boundaries, we leverage unsupervised style transfer to achieve strong performance in weather and lighting conditions that deviate substantially from those of the annotated datasets that are publicly available. We evaluate our approach on the TuSimple lane marking challenge and difficult CULane datasets [23] and achieve competitive accuracy while running at 40 FPS.

1. Introduction

Lane detection is as fundamental to self driving as it is imperative. Unlike dynamic scene which is occasionally irrelevant or nonexistent, static scene - aspects of the environment that change on the time scale of months or longer - informs a self-driving car's notion of state at every time step. Instead of relying on direct perception to infer these semantic labels, a popular approach for modern self-driving teams today is to leverage *high definition maps* of the environment for a dense, reliable estimate of static scene. The perception of these important semantic features is thereby reduced to a localization problem which technology such as G.P.S., lidar, I.M.U. and visual odometry can solve to centimeter accuracy [28].

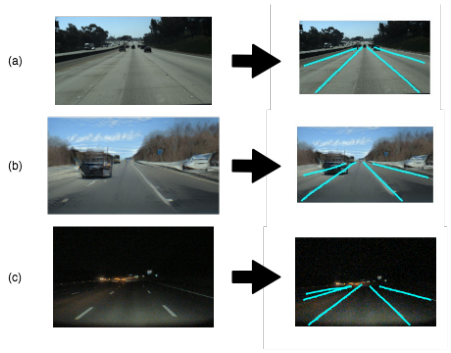

Figure 1. Best viewed in color. We train a novel end-to-end lane detection network on a public dataset of labeled sunny California roads (a). We use style transfer to associate annotations with pseudo massachusetts images and retrain (b). Finally, we evaluate the network in diverse weather conditions on real images from Massachusetts and find the network handles the "long tail" (c).

Despite the successes of high definition maps, lane detection remains an important possible solution to a variety of tasks relevant to autonomous vehicles. For cases where high definition maps are flawed or unavailable, direct, reliable perception of lanes remains an unsolved and crucial task. [11] shows that accurate offline lane detection can streamline the process of generating high definition maps. For systems such as driver assist programs that are not equipped to handle the computational load of effective usage of high definition maps, online lane detection remains the central problem to be solved. If direct perception of lanes can be made reliable, the overhead of robust self-driving induced by the dependence on high-definition maps will be dramatically reduced. In pursuit of solving these problems, 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. Variability in the number of instances of an object in an image is by definition an aspect of detection. Variability in represen-

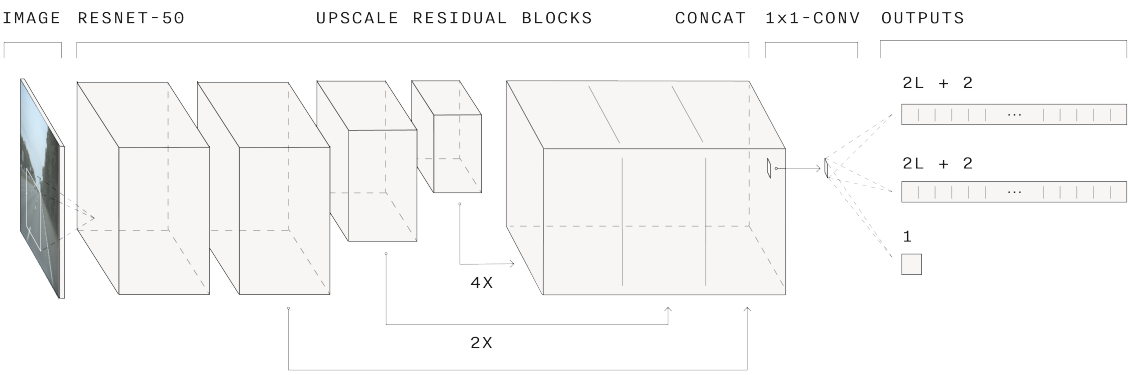

Figure 2. We use a convolutional encoder to extract semantic features of an input image. These features are then decoded by three separate convolutional heads: a mask of pixels in the image that belong to a lane, and a categorical distribution over the pixels within a distance L of the current pixel in the rows above and below.

tation is more unique to the lane detection problem. Unlike bounding boxes which have a precise encoding of fixed dimensionality, lane segments are potentially arbitrary length. Representations of lanes such as polynomials lose accuracy on tight curves where accurate lane detection is most important.

Second, the detection algorithm must run in real-time in order to be useful for self-driving. Therefore, although there is variability in the number and size of lanes in an image, whatever recursion or ranking is used to identify and draw these lanes must be simple. Solutions to the variable dimensionality problem that involve recurrent cells [10] or attention [27] are therefore ideally avoided.

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 these cases in a scaleable 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 and requires only the simplest possible recursion. Our network comes equipped with a natural definition of uncertainty. It's performance is competitive with El-GAN and [8] and CULane [23] and runs at 40 frames per second on a GTX 1080.

- 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. While style transfer has been used extensively to adapt the output distribution of simulators to better match reality, 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 Solutions to the lane detection problem 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 as demonstrated by [2] and [14]. 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. The authors of [23] rephrase lane detection as semantic segmentation where categories are left-left lane, left lane, right lane, and right-right lane. [22] uses a CNN to do instance segmentation of lanes and to predict a homography that simplifies and regularizes curve fitting of individual lanes. [8] incorporates an adversarial loss to force a segmentation network to match the plausible lane label distribution for curves.

Lane detection is not isolated to dashcam imagery. [11], [17], and [4] leverage data from lidar point clouds, open street maps, satellite imagery to detection lanes. 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 such as [5], [20], and [19].

Style Transfer Adversarial loss has recently enabled rapid growth in a wide range of supervised and unsupervised tasks. Pix2Pix [13] was the first to demonstrate success in style translation tasks on images. Unsupervised style transfer for images [21] [31] and unsupervised machine translation [18] [3] use back-translation as a proxy for su-

pervision. While [21] and [31] learn deterministic translators, the authors of [12] build on the work of [21] to generate a distribution of possible image translations. In this work, we incorporate images generated by [12] translated from a public dataset to match the style of our own cameras. We choose [12] for this purpose because it is unsupervised and generative. Our work has similar goals to [30] which uses GTA-V to train object detectors that operate on lidar point clouds. We seek to leverage human annotated datasets instead of simulators as seeds for generating pseudo training examples of difficult environmental conditions.

Other We take are inspired by work in human pose estimation [6] and automated object annotation [1]. [6] solves the association problem between segmentations of human body parts by inferring slope fields between body parts that are part of the same human. 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 [7] and Sketch-RNN [9], 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 realtime performance.

3. Model

In the semantic segmentation formulation of lane detection, we train a model to estimate the probability that each pixel in an input image $\mathbf{x} \in \mathbb{R}^{3\times H\times W}$ belongs to a lane. If f is a fully convolutional neural network, the probability of label $c \in \{0, 1\}$ at pixel x_{ij} is parameterized by

$$p(c|x_{ij}) = \text{Softmax}(f(\mathbf{x})_{ij}). \tag{1}$$

In general, a decoder that takes the output of $f(\mathbf{x})$ as input is used to cluster the pixels into individual lane boundaries. [22] uses operations on the encodings of pixels to estimate the interaction between pixels. Other approaches decode using RANSAC [2]. These approaches are in general heuristic-based, can be time instensive, and do not pass gradients to the encoder f.

3.1. End-to-End Lane Detection

We parameterize decoding by inferring the probability pixels local to each other share the same lane label. Let $\mathbf{y} = \{y_1, ..., y_n\}$ where $y_i \in \mathbb{R}^2$ be waypoints on an annotation of a lane in pixel coordinates. The probability $p(\mathbf{y}|\mathbf{x})$ can be factored

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

One choice to predict $p(y_i|y_1, ..., y_{i-1}, \mathbf{x})$ would be to use a recurrent neural network. [11] and [29] use recurrent neural networks successfully for this task. Do decode quickly, we assume most of the dependency can be captured by conditioning only on the previous waypoint

$$p(\mathbf{y}|\mathbf{x}) \approx p(y_1|\mathbf{x}) \prod_{i=2}^{n} p(y_i|y_{i-1}, \mathbf{x}). \tag{3}$$

Lane detection is then reduced to predicting the probability each pixel belongs to a lane $p(y_1|\mathbf{x})$ and the probability a pixel belongs to a lane conditioned on a nearby pixel belonging to a lane $p(y_i|y_{i-1}, \mathbf{x})$.

To represent the distribution $p(y_i|y_{i-1}, \mathbf{x})$, 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. Inspired by [26], we choose to make no assumptions about the shape of $p(y_i|y_{i-1}, \mathbf{x})$ and represent the pairwise distributions using categorical distributions. At each pixel $x_{i,j}$, our network predicts

- $p_{i,j,0} = p(x_{i,j})$ - the probability that pixel $x_{i,j}$ is part of a lane.
- $p_{i,j,1} = \{p(x_{i+1,j+l}|x_{i,j})\}_{-L<=l<=L} \cup p(\text{end}|x_{i,j})$ - the categorical distribution over pixels in the row **above** pixel i, j within a distance L that pixel $i+1, j+l$ is part of the same lane as pixel i, j, or that pixel i, j is the final pixel in the lane it is a part of.
- $p_{i,j,-1} = \{p(x_{i-1,j+l}|x_{i,j})\}_{-L<=l<=L} \cup p(\text{end}|x_{i,j})$ - the categorical distribution over pixels in the row **below** pixel i, j within a distance L that pixel $i-1, j+l$ is part of the same lane as pixel i, j, or that pixel i, j is the final pixel in the lane it is a part of.

Given these probabilities, we can quickly decode a full lane segment given any initial point on the lane. Given some initial position x_0, y_0 on lane l, we follow the greedy recursion

$$l(x_0) = y_0 \tag{4}$$
$$l(x + sign) = l(x) + \Delta x \tag{5}$$
$$\Delta x = -L + \text{argmax} p_{x,l(x),sign} \tag{6}$$
$$\tag{7}$$

where $sign \in \{-1, 1\}$ depending on if we draw the lane upwards or downwards from x_0, y_0. We stop decoding when argmax returns the `end` token. Since our network predicts the full distribution over the next point in the lane, we could use sampling based methods to approximate the optimal subset of pixels that form the lane. We choose a greedy decoder for the sake of speed.

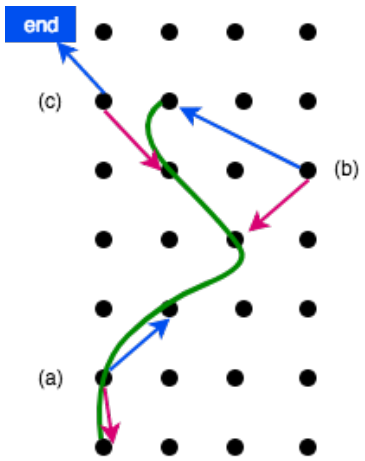

Figure 3. In addition to binary segmentation, we train our network to output the pixel in the row above and below that is in the same lane as the current pixel. Shown are ground truth up and down labels for three pixels with respect to an annotation in green. In addition to training points that are members of the annotated lanes themselves (a), we address exposure bias by also training points that are offset from annotated lanes (b). We include the `end` token in the categorical distribution which signals 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.

3.2. Loss

We minimize the negative log likelihood given by (3). Let θ represent the weights of our network, $\mathbf{x} \in \mathbb{R}^{3,H,W}$ an input image, $\mathbf{y} = \{\{x_1, y_1\}, ..., \{x_n, y_n\}\}$ a ground truth lane annotation such that $y_i - y_{i-1} = 1$ and $x_i, y_i \in \mathbb{N}$, and $\mathbf{y_m} \in \mathbb{R}^{2,H,W}$ a ground truth segmentation mask of the lane. The loss $L(\theta)$ is defined by

$$L_{bin}(\theta) = -\log(p(\mathbf{y_m}|f_\theta(\mathbf{x}))) \tag{8}$$

$$L_{pair}(\theta) = -\sum_{i=2}^{n} \log(p(\{x_i, y_i\}|\{x_{i-1}, y_{i-1}\}, f_\theta(\mathbf{x}))) \tag{9}$$

$$L(\theta) = L_{bin}(\theta) + L_{pair}(\theta) \tag{10}$$

Because the the task of binary segmentation and pairwise prediction have different uncertainties and scales, we choose to use [15] to dynamically weight these two objectives. We incorporate a learned temperature σ which is task specific to weigh our loss:

$$L(\theta) = \frac{1}{\sigma_{bin}^2} L_{bin}(\theta) + \frac{1}{\sigma_{pair}^2} L_{pair} + \log \sigma_{bin}^2 \sigma_{pair}^2. \tag{11}$$

During training, we substitute $W = \log \sigma^2$ into (11) for numerical stability. We note that empirically, 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 experimental results reported below to avoid tuning hyperparameters.

3.3. Exposure Bias

If the loss $L_{pair}(\theta)$ is evaluated exclusively on ground truth lane boundaries, our network will experience significant exposure bias due to the fact that the pixel we condition on will always be truly part of a lane boundary. 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 [25] and achieve good results.

Although we experiment with reinforcement learning, we find that training the network to denoise lane annotations is significantly more sample efficient. 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 therefore generate training examples for $sign \in \{-1, 1\}$

$$s \sim \lfloor \mathcal{N}(0.5, \sigma \mathbf{I}) \rfloor \tag{12}$$

$$p(i, j + s, sign) = l(i + sign) - j - s + L. \tag{13}$$

We tune σ as a hyperparameter which is dependent on dataset and image size. We clamp the ground truth difference $l(i + sign) - j - s + L$ at 0 and $2L + 1$ and $j + s$ by 0 and the width of the image.

3.4. Adaptation

An ideal lane detection network would be able to perform at high accuracy in all environments. The downside of data-driven approaches is that we have weaker 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 [12] 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 [31] 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.

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

$$\mathbf{x}, \mathbf{y} \sim D \tag{14}$$

$$\mathbf{x}' \sim G'(E(\mathbf{x}, s))_{s \sim \mathcal{N}(0, \mathbf{I})} \tag{15}$$

$$\mathbf{y}' = \mathbf{y} \tag{16}$$

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 [23]. 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 [8]. 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 [23].

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 bias across all images to develop D'.

$$y_i' = m y_i + b \tag{17}$$

Random samples from the pseudo training example generator are shown in figure 4. We compare qualitatively the results of a network trained on D and D' versus the lane detection module on the NVIDIA PX2.

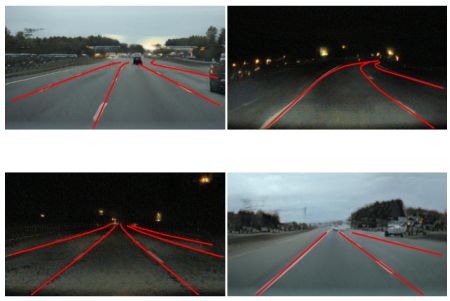

Figure 4. Four samples from D' are shown. Images are generated by the style transfer network and lane annotations are copied from the annotations associated with the Tusimple image that was translated. The style translation preserves lane boundaries allowing us to provide FastDraw a training signal which is more robust and general than the training signal provided by samples from purely the Tusimple Dataset.

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 [16]. 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 [24] 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 [8]. 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 finish the instance segmentation. FastDraw is competitive with El-GAN by all Tusimple metrics as shown in Table 1. We resize the input Tusimple images by half to train and evaluate FastDraw, and achieve best results with $L = 16$.

We carry out an ablation experiment on the extent to which input images are resized during training and evaluation and report scores in Table 2. We find that the model performs just as well if not better in certain metrics on images of size 144 x 256. For models trained on images resize by 0.2, we set $L = 6$.

5.2. Uncertainty

Because our network predicts a categorical distribution at each pixel with domain $-L <= l <= L$, we can quickly compute the standard deviation of this distribution and in-

Method	Accuracy (%)	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 (D')	**95.2**	0.076	**0.045**

Table 1. We use Tusimple evaluation metrics to compare quantitatively with [8]. FastDraw achieves comparable performance to EL-GAN with fewer layers. We note that while the accuracy of FastNet trained on D' in addition to D 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.

Method	Resize	Accuracy (%)	FP	FN
Resnet18	0.2	94.7	0.054	0.055
Resnet50	0.2	94.8	0.053	**0.050**
Resnet18 (D')	0.2	94.7	**0.049**	0.057
Resnet50	0.5	**94.9**	0.059	0.052

Table 2. We evaluate the ability of the FastDraw framework to perform on small images. We find that training and evaluating on images that are resized by 0.2 to a size of 144 x 256 does not hamper the performance of the FastDraw decoder. Our smallest model - Resnet 18 with resize 0.2 - trains for 7 epochs in 10 minutes and achieves competitive performance with El-GAN [8].

terpret 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 5. These estimates can be propagated through the self-driving stack to control the vehicle safely 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 simple 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. This trend suggests that the accuracy improvement of FastDraw over `argmax` style decoding used as a baseline in [8] is due to the ability of the network to learn its own prior over lane geometry.

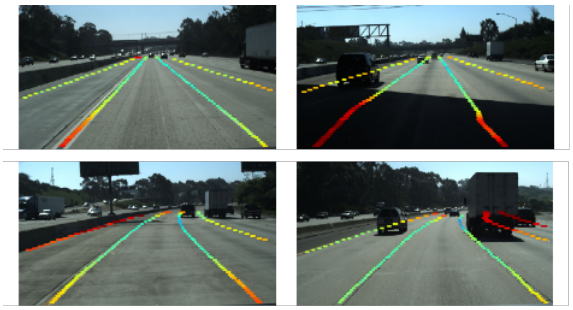

Figure 5. The standard deviation of the distribution predicted by FastDraw is plotted as error bars on a variety of images from Tusimple test set. Our color map is thresholded at a standard deviation of 0 and 9 pixels. We find that the network is accurately growing more uncertain in regions where the 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.

Distance to heuristic	%
$d < 1$	12.6
$d < 3$	58.2
$d < 5$	87.1

Table 3. We calculate the distance $d = |p(i,j,1) - \text{argmax}_{j-L<=l<=j+L} p(i+1, j+l, 0)|$ over pixels (i,j) that are part of a lane predicted by FastDraw. 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 $p(i,j,0)$ predictions. Deviation from the naive decoder explains why the network still performs well when the segmentation mask is noisy, as in row (f) of 6

5.4. CULane

We train FastDraw on the full CULane training dataset [23]. 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 find 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 [23] 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 waypoint geometry of lanes.

5.5. Massachusetts

We evaluate the ability of FastDraw to generalize to new scenes. In figure 6 we demonstrate 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

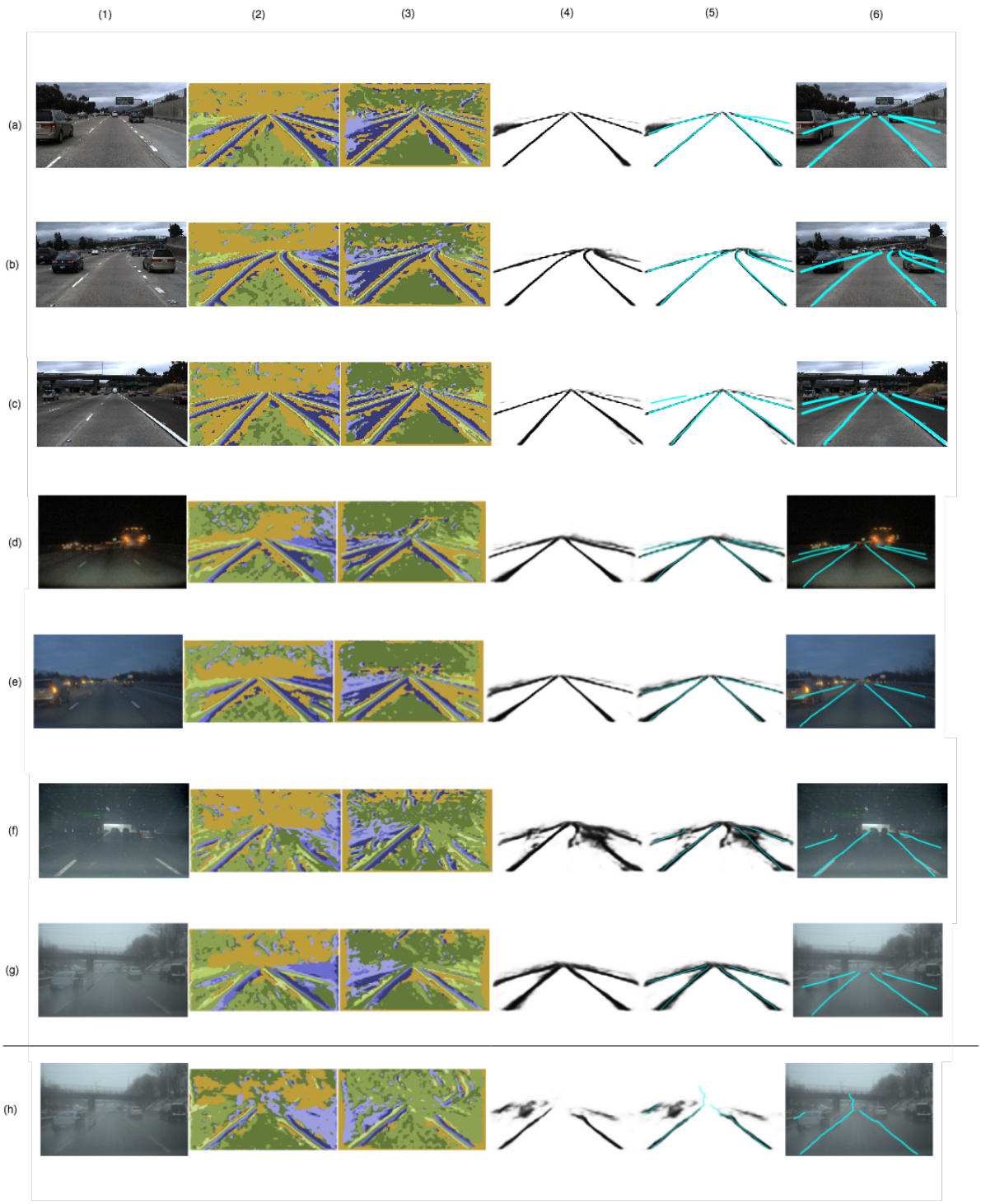

Figure 6. We demonstrate that our network can perform with high accuracy on the publicly annotated dataset we train on as well as images collected from a diversity of weather conditions and scenes in Massachusetts. We visualize the output of FastDraw Resnet50 (D'). (1) shows the input image, (2) shows the argmax of the $p(i, j, 1)$ head, (3) shows the argmax of the $p(i, j, -1)$ head, (4) shows the output of the $p(i, j, 0)$ head, (5) shows the decoded lane predictions on top of the $p(i, j, 0)$ output, and (6) shows the decoded lane predictions on top of the input image. Rows (a) - (c) show performance on samples from the Tusimple test set. Rows (d) - (g) demonstrate performance on challenging images taken from dash cam footage around Massachusetts. (h) shows output of FastDraw trained only on Tusimple training set evaluated on the same Massachusetts image as (h). The model performs remarkably well on both clean and noisy datasets without the need for extra human annotation. Even when the segmentation $p(i, j, 0)$ is noisy as in (f), or evaporates as in (c), the decoder accurately decodes the lanes.

	ResNet-50 [23]	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 [23], we emphasize that architectural improvements such as those introduced in [23] will likely complement performance of the FastDraw decoder.

that no additional human annotation was required to train FastDraw to be robust to these difficult environments.

Visualization of the output of FastDraw on these difficult scenes when it is not trained on the style transfer dataset shows that all output heads of the network do not generalize. We visualize the output of FastDraw trained on D and compare it to the output of FastDraw trained on $D \cup D'$ in rows (g) and (h) of Figure 6. We find that all three heads of the network benefit from the generated training examples produced by style transfer.

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 assumptions 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. 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. 2

[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. 2

[5] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015. 2

[6] 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

[7] L. Castrejón, K. Kundu, R. Urtasun, and S. Fidler. Annotating object instances with a polygon-rnn. *CoRR*, abs/1704.05548, 2017. 3

[8] M. Ghafoorian, C. Nugteren, N. Baka, O. Booij, and M. Hofmann. El-gan: Embedding loss driven generative adversarial networks for lane detection, 2018. 2, 5, 6

[9] D. Ha and D. Eck. A neural representation of sketch drawings. *CoRR*, abs/1704.03477, 2017. 3

[10] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997. 2

[11] 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, 2, 3

[12] X. Huang, M. Liu, S. J. Belongie, and J. Kautz. Multimodal unsupervised image-to-image translation. *CoRR*, abs/1804.04732, 2018. 3, 4

[13] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016. 2

[14] 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

[15] 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. 4

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 5

[17] A. Laddha, M. K. Kocamaz, L. E. Navarro-Serment, and M. Hebert. Map-supervised road detection. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 118–123, June 2016. 2

[18] G. Lample, L. Denoyer, and M. Ranzato. Unsupervised machine translation using monolingual corpora only. *CoRR*, abs/1711.00043, 2017. 2

[19] 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. 2

[20] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. 2

[21] M. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. *CoRR*, abs/1703.00848, 2017. 2, 3

[22] 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, 3

[23] 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, 6, 8

[24] 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. 5

[25] 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

[26] 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

[27] 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

[28] G. Wan, X. Yang, R. Cai, H. Li, H. Wang, and S. Song. Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes. *CoRR*, abs/1711.05805, 2017. 1

[29] Z. Wang, W. Ren, and Q. Qiu. Lanenet: Real-time lane detection networks for autonomous driving, 2018. 3

[30] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud, 2018. 3

[31] 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. 2, 3, 4