Pavement crack detection

Professor: Volker Rodehorst
Student: Khanh Ha

Paper collection

- Collected about 26 papers regarding crack detection

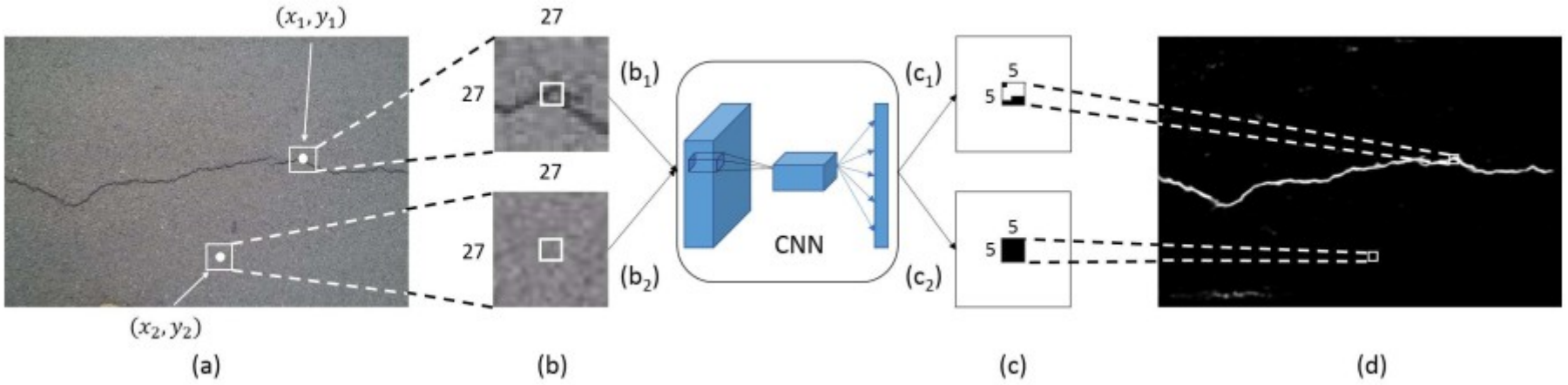
Pick a paper for implementation

- Automatic Pavement Crack Detection Based on Structured Prediction with the Convolutional Neural Network.
- Why?
  - the data set is available
  - the main idea of the paper is understandable to me.
  - Just want to pick a simple one to quickly get an understanding about the problem

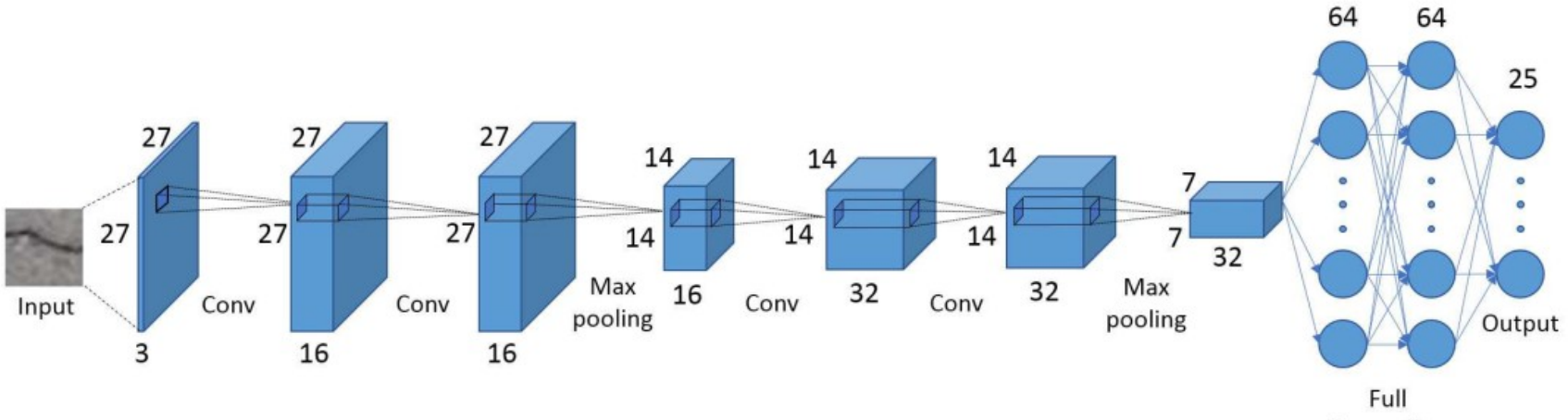
Data set

- The CFD database contains 118 RGB images with a resolution of 320 × 480 pixels
  - Images are taken from iphone 5 from pavements of Beijing.
- The AigleRN database contains 38 gray-level images
  - Images been collected on French pavement
  - Pre-processed to reduce non-uniform illumination

Main idea



Architecture



Input	→	conv16 [3×3,1,1]	→	conv16 [3×3,1,1]	→	maxpool [2×2,2,0]	→	conv32 [3×3,1,1]	→	conv32 [3×3,1,1]	→	maxpool [2×2,2,0]	→	FC64	→	FC64	→	FC25

Architecture

- Cross entropy loss

$$L = -\sum_{i=1}^{s^2} (y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i))$$

Architecture

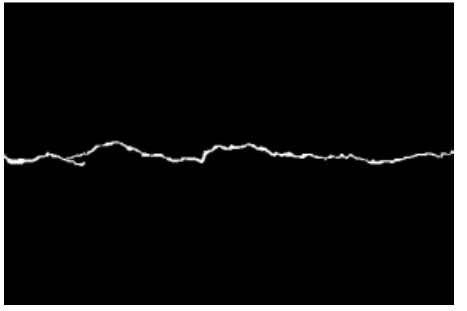
- Over-fitting reduction

$$L' = L + \beta \cdot \frac{1}{2} \sum_j W_j^2$$

- Dropout layer: 0.5
- Batch normalization layer: added by me

Implementation: data extraction

- Positive patch extraction (crack patch)
  - Skeletonize the ground true mask to shrink the crack.
    - I think that doing it would help reduce redundancy
  - Extract patch of [27,27] at each crack pixel.
- Negative patch extraction
  - Dilate the ground true mask: dilation with = 13
  - Extract patch of [27,27] at each black pixel

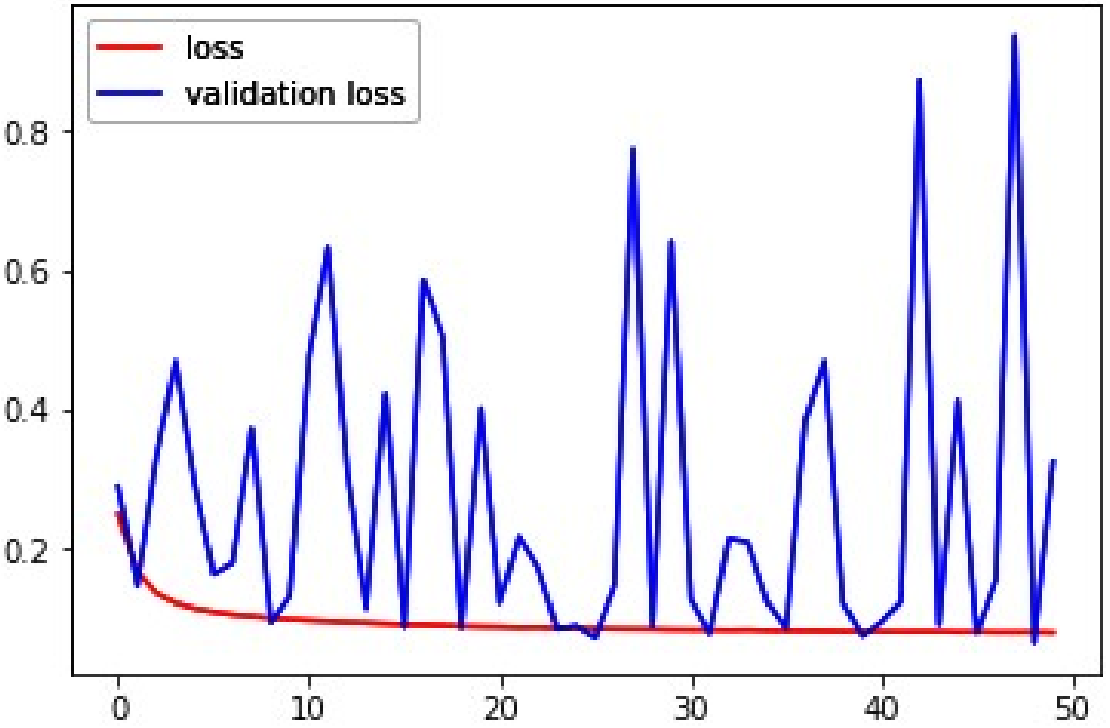


Implementation: training process

- Split patch list (1.3 million positive and negative patches)
  - Testing: 25%
  - Training: 55%
  - Validation: 20%
- Batch size: 256
- Epochs: 50
- Weight initialization: xavier method
- After an epoch ends
  - Shuffle the training patch list

Implementation: learning curve

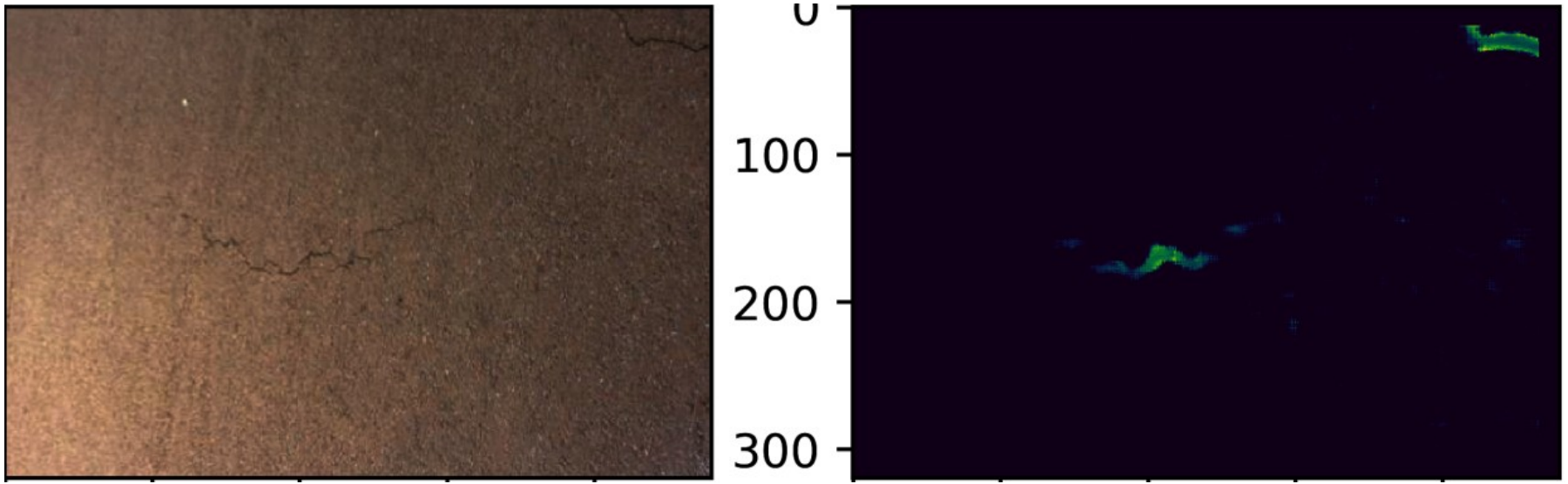
- Why is there noise in the validation loss?



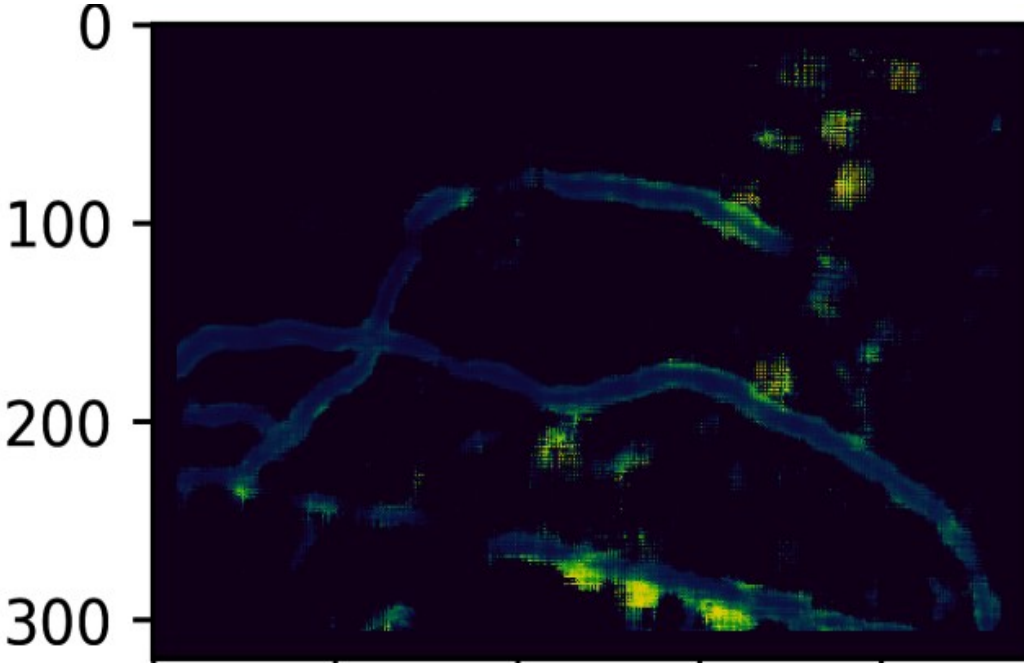
Implementation: accuracy

- Test loss: 0.0636
- Test accuracy: 97.271

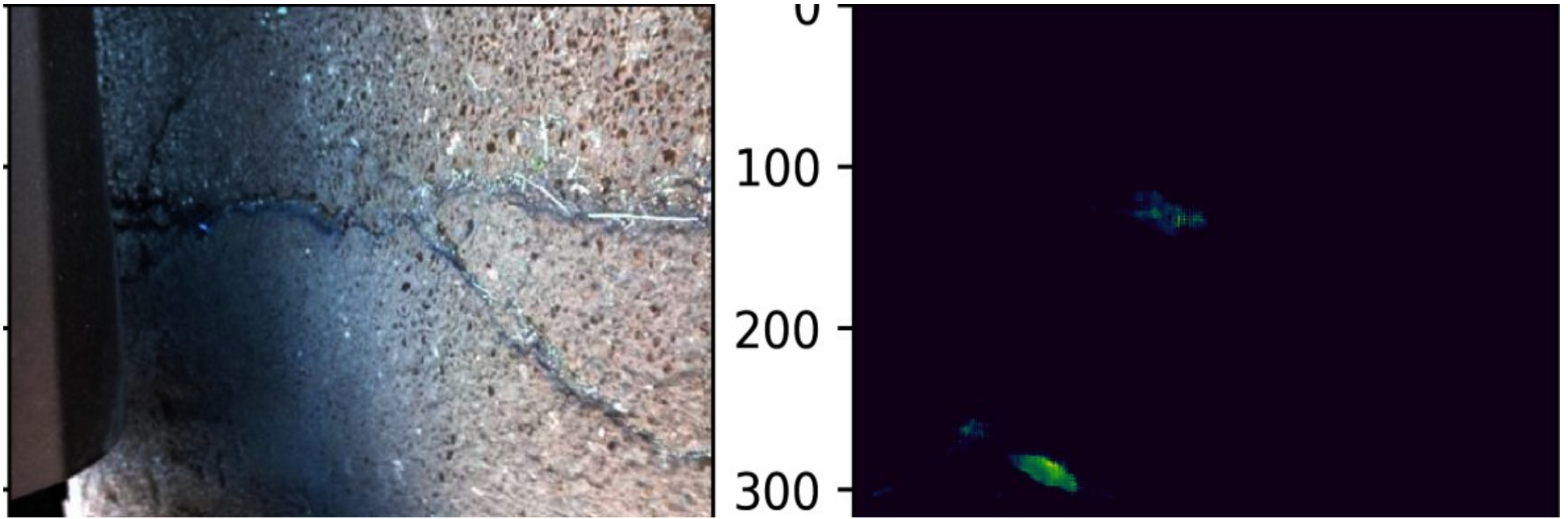
Inference result



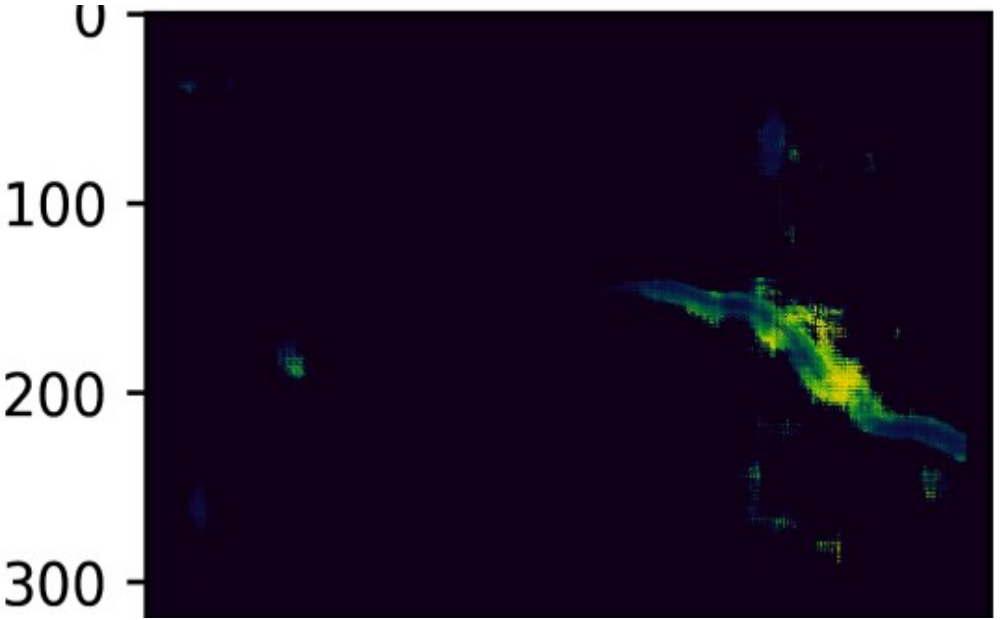
Inference result



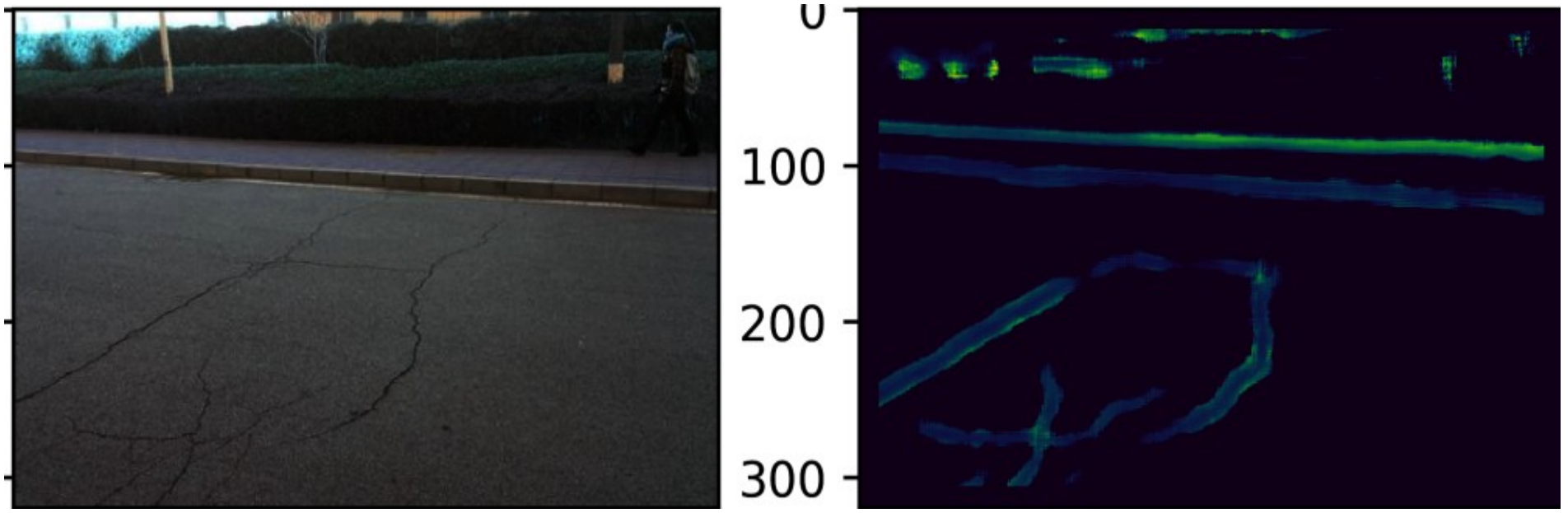
Inference result



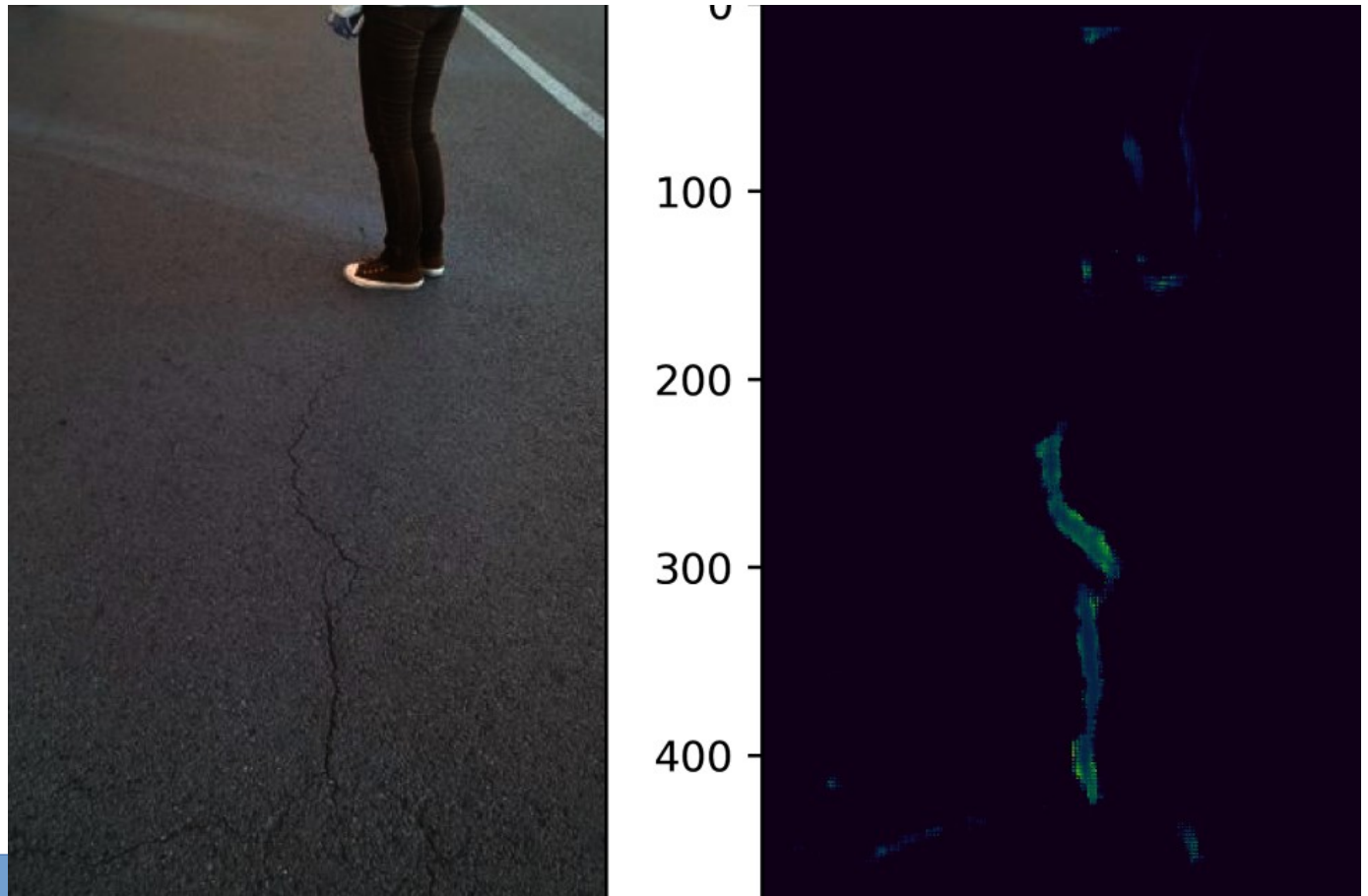
Inference result



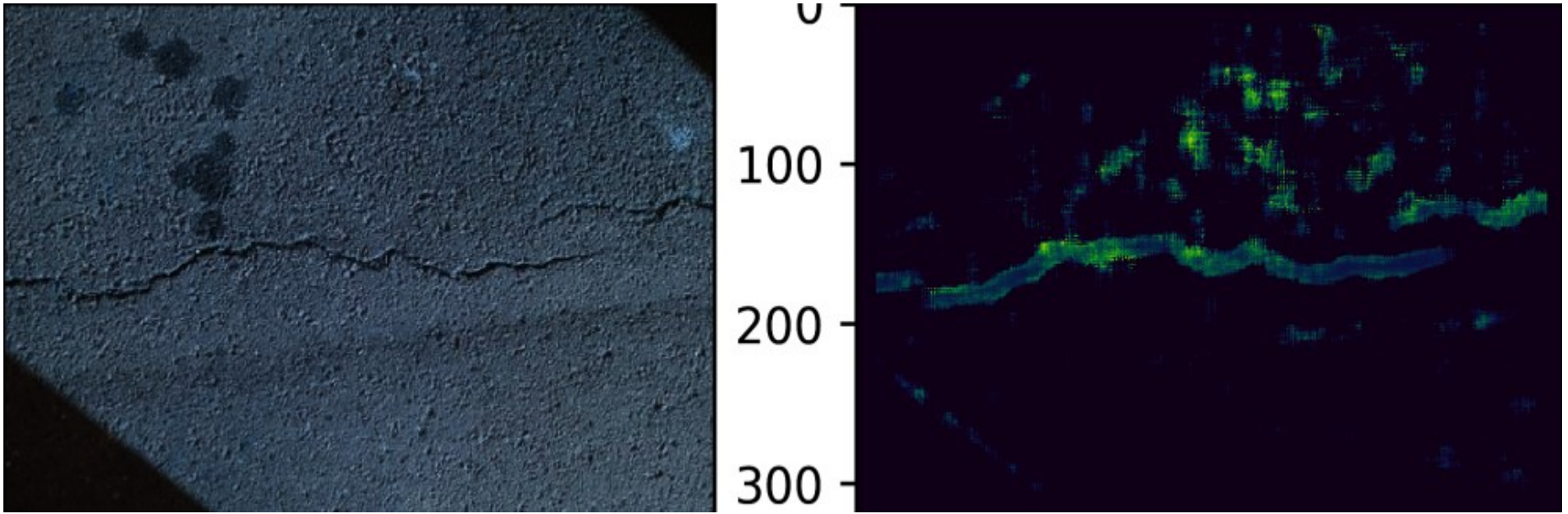
Inference result



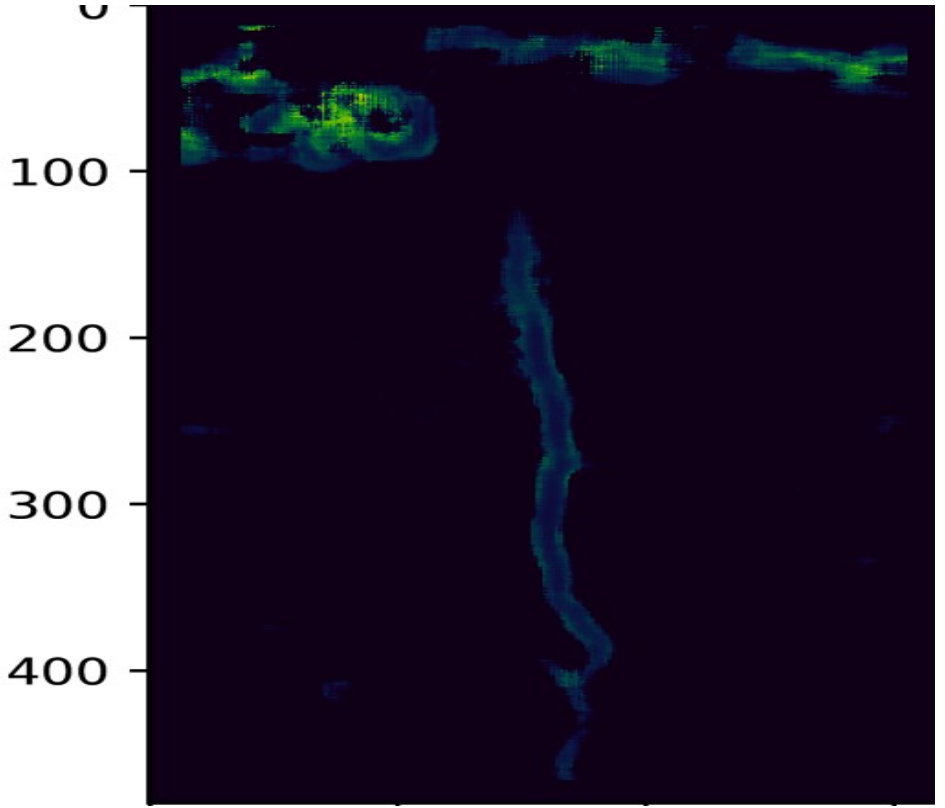
Inference result



Inference result



Inference result

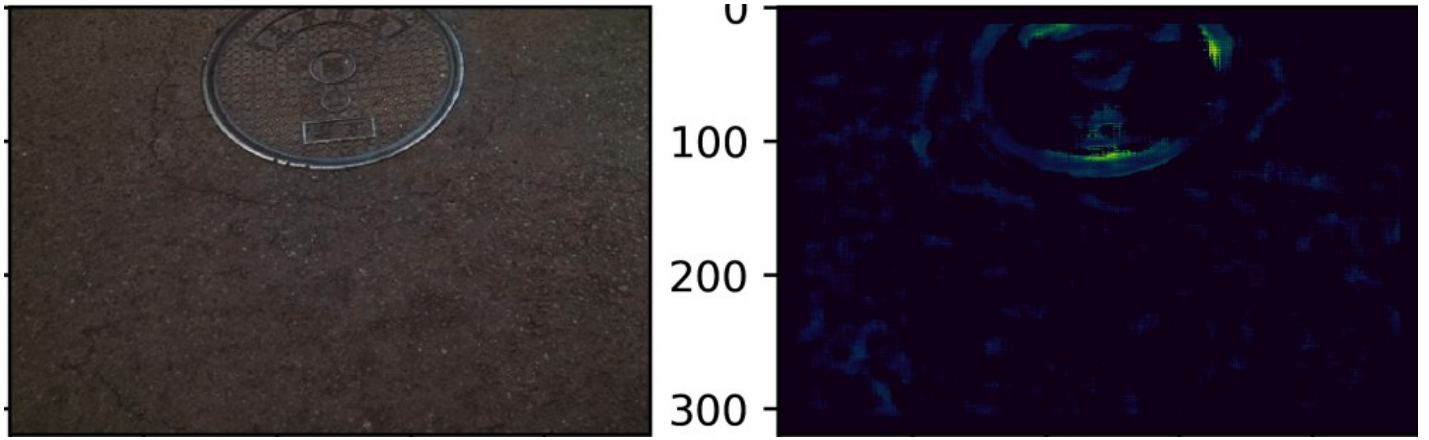


observation

- Is the current result better than canny edge?
  - NO

Need more global information

- A patch of [27,27] is small
  - doesn’t give model enough semantic information about a crack
  - It recognizes this one as crack



Imbalanced data

- The accuracy is high because of the imbalanced data
  - How to check it
    - Needs to find out out how calculate recall/precision/F1 score given the current result

what’s next?

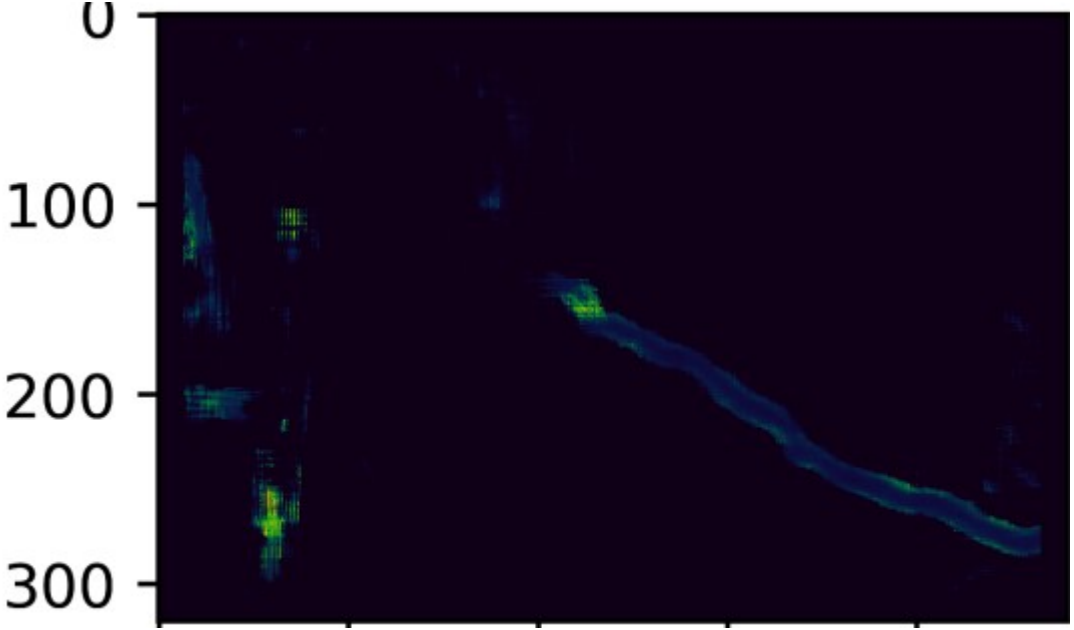
- define clear the problem of the current approach
- transfer learning ?
- data augmentation ?
- look for ideas from other paper

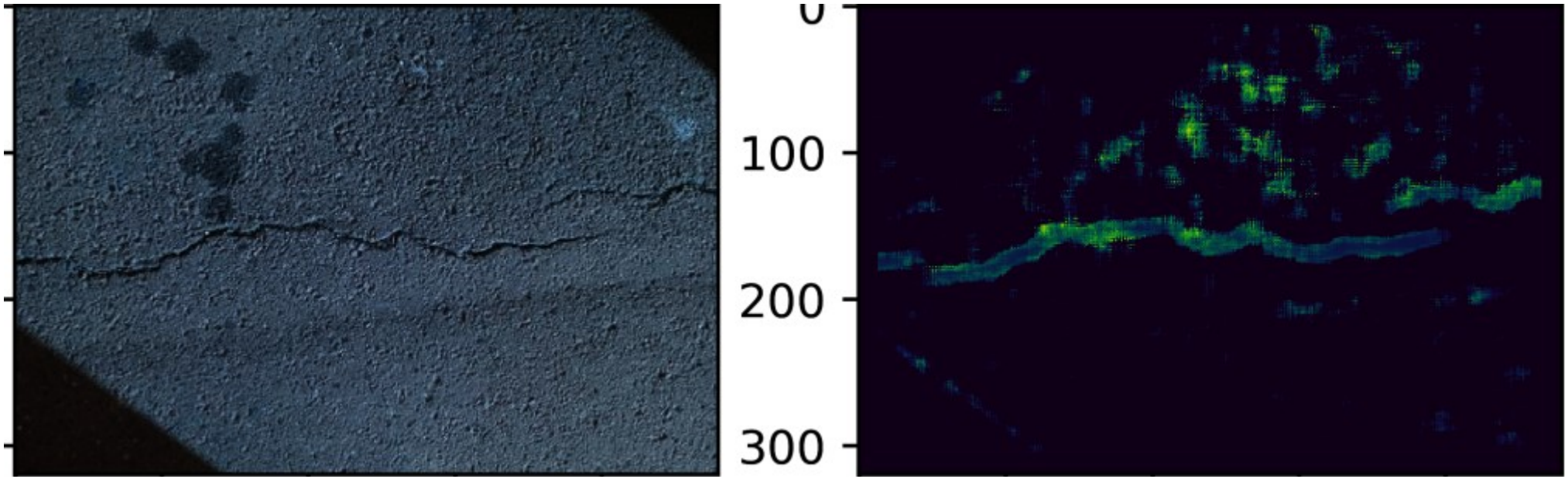
Takeaway

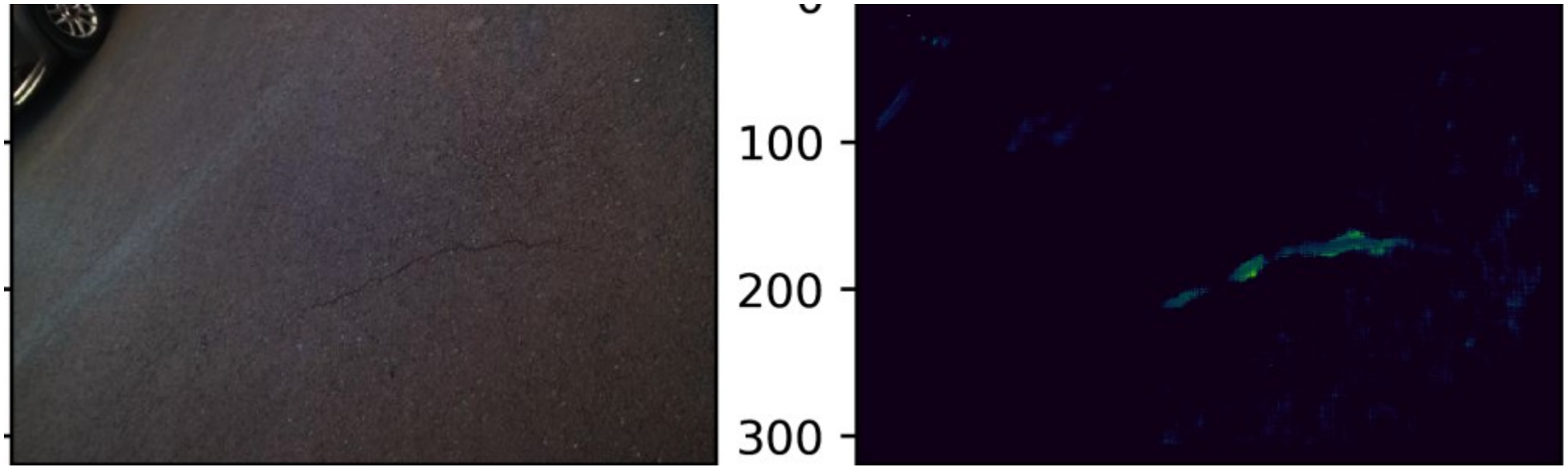
- Remember to normalize data at the inference step
  - It took me at least several hours to figure out why the model prediction is just **0**
  - **Convert: [0 255] => [0 1]**

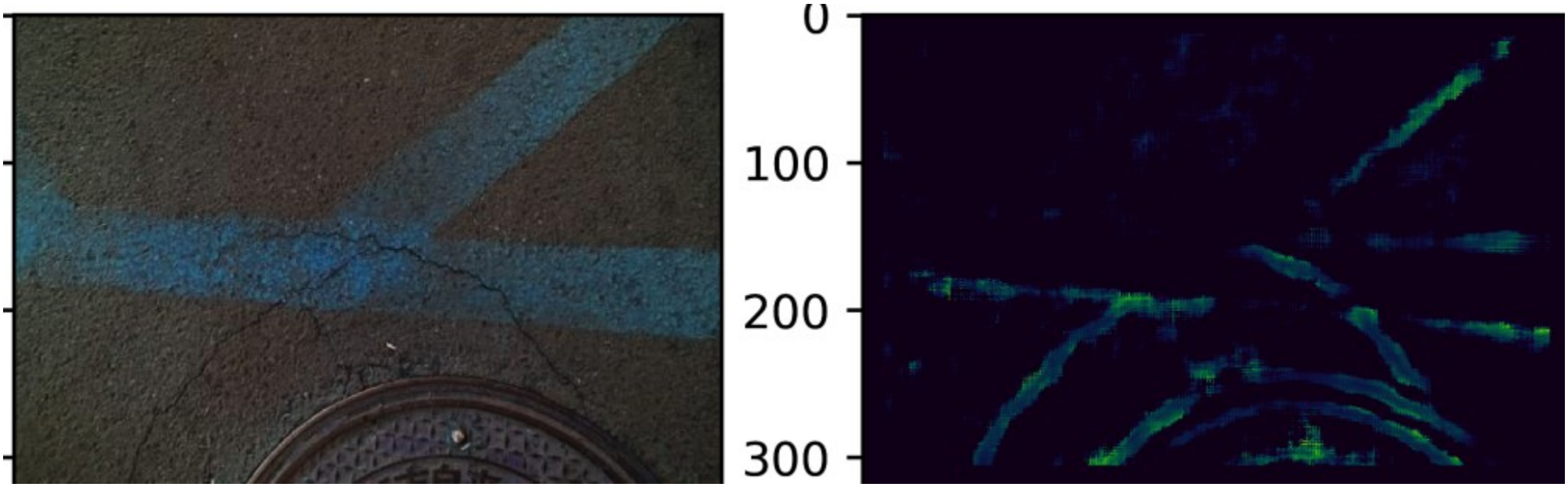
Thank for your attention!

More result




0
100
200
300


100
200
300


0
100
200
300