Bare Demo of IEEEtran.cls for IEEE Journals

Michael Shell, *Member, IEEE,* John Doe, *Fellow, OSA,* and Jane Doe, *Life Fellow, IEEE*

***Abstract*—The abstract goes here.**

***Index Terms*—IEEE, IEEEtran, journal, LaTeX, paper, template.**

I. Introduction

M: Motivate need for powerline inspection

Electricity is the lifeblood of modern society, thus, it is essential to ensure stable electrical power supply across the nations. Hence, it is of utmost importance for utilities to inspect and maintain their electrical facilities on a regular basis. These tasks were conventionally done by human inspector manually following, observing and assessing the power grid. Recently, Unmanned Aerial Vehicles (UAVs) have been deployed to produce more quality and efficient observations due to their superior speed and ability to access higher altitude and more hazardous environments. To further boost the effectiveness and efficiency of inspection and maintenance tasks, the assessment step can be done automatically from the observations of UAVs, which, in many cases, are RGB images from cameras. Many automatic assessment solutions have recently been proposed, many of which are fueled by the power of artificial intelligence and deep learning. Examples are (insulators, transmission towers, defects, severity). Among these tasks, cable, wire or powerline recognition and localization is very crucial. Accurate detection of powerlines helps focus the scope of downstream solutions for diagnosing faults (Tears, kinking, bird caging, etc...) and identifying intrusions (vegetation encroachment, etc,) on said powerlines. These tasks are extremely vital as failure in catching these undesirable events can lead to catastrophic and fatal consequences [**?**]. Furthermore, the ability to detect powerlines is imperative for UAVs, and other forms of low-altitude flights, to navigate safely.

However, it is nontrivial to detect powerlines due to their inconspicuous appearances. The powerline can be very thin and might be missed by approximity sensors on UAVs. On camera images, the width of powerlines can be as thin as 1 pixel. Furthermore, cluttered backgrounds and hindered powerline visibility and same-colored background, such as snow and white coating, can cause the powerlines to be imperceptible. Other conditions, such as fog and lighting, can also negatively affect the discernability of powerlines on images.

M: Current solutions leverage DL

M: Current state-of-the-art: Motivate shortcomings. Overlapping lines, can include a picture of it to motivate it further.

M: In this work, ... (we propose the solution).

M. Shell was with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332 USA e-mail: (see http://www.michaelshell.org/contact.html).

J. Doe and J. Doe are with Anonymous University.

Manuscript received April 19, 2005; revised August 26, 2015.

II. Related Work

The problem of powerline detection has been tackled with traditioncal computer vision and more recently with deep learning approaches. An notable early attempt was proposed by Kasturi et al. [**?**], where Steger's method [**?**] was used to extract features, which are then filtered using the Hough transform to exclude short lines. Yan et al. [**?**] proposes **M: to leverage the?** Radon transform to generate line segments, group these segments by slope and distance thresholding, and then finally use Kalman filters as a post-processing step. Li et al. [**?**] removes clutter and noise in the background using a pulse coupled neural filter before using Hough transforms and K-means clustering to detect and combine line segments. The aforementioned solutions and some others [**?**], [**?**], [**?**], [**?**] hold the strict assumptions that powerlines appear straight and in parallel orientations, and thus, do not always apply in reality. Song et al. [**?**] was able to detect curved powerlines by using a normalized graph cut model to link line segments, which were produced from the responses of a match filter and first-order derivative of a Gaussian. However, the performance of all the methods above are affected by the conditions that the input images were taken in. Any differences in camera settings, environment, lighting, and view angle can lead to the hyper-parameters of these methods to be re-defined, which in turn may requires specialist expertise.

Overall, deep learning can help alleviate the problems that traditional computer vision face. With enough data, deep learning solutions for classification and detection can generalize well to many practical scenarios. Pan et al. [**?**] suggests training a convolutional neural network (CNN) that takes in edge features, which are produced by steerable filters, and classifies whether square patches from an input image contains a line or not. Then the Hough transform is used to detect the powerline segments. Similarly, Gubbi et al. [**?**] also proposes using a CNN but, instead, uses Histogram of **M: Oriented** Gradient features as input. The line segment detector [**?**] was used as a post processing step. Since the inputs of these two methods are individual patches, the CNN models lack the contextual information of the entire images. Hence the performance can be limited, especially for low-contrast images with cluttered background. More recent approaches [**?**], [**?**], [**?**], [**?**], [**?**], [**?**] frame powerline detection as binary pixel-level

classification problems where each pixel is classified whether is belongs to the powerline or not. This type of problem can be undertaken by deep learning semantic segmentation networks. **M: Madaan et al. [?] investigate different dilated convolutional neural networks for segmenting powerlines.**

Yetgin et al. [**?**] finetuned a CNN, which was pretrained on ImageNet [**?**], with a newly initialized softmax layer that classifies the powerline existence. After that, features from immediate layers of the finetuned CNN are used to train another classifier to perform powerline segmentation. In [**?**], a CNN is introduced with two components: an information fusion module and an attention module. The information fusion module is in encoder-decoder structure, where decoding stages are combined with their corresponding same-scaled encoding stages to fuse semantic and location information for accurate powerline segmentation. **M: Might make sense to briefly comment on the attention module here as well as you say it consists of two components but then just detail one of them.** Abdelfattah et al. [**?**] trained a generative adversarial network (GAN) to generate modified versions of the input images where the powerlines are highlighted. A semantic decoder connected to an immediate layer of the generator is also trained jointly in order to perform the actual segmentation task. These semantic segmentation methods have achieved satisfying results, however, they require training data with pixel-level annotation, which can be laborious to obtain. Especially in the case of powerlines, which can often be quite slender yet span across images, more meticulousness is required in the annotating process. Choi et al. [**?**] tries to mitigate this burden by approximating pseudo segment maps, which are produced using the Visualbackprop algorithm [**?**] from a classification CNN. This classification CNN is trained only with patch labels. Lee et al. [**?**] on the other hand trained a CNN on images with only image-based labels. During inference, this CNN is used on patches of high-resolution images and produces feature maps which can be combined for powerline localization.

M: Provide high-level overview of current powerline inspection approaches and LS-Net. Mention shortcomings

III. Methodology

A. Preliminaries: LS-Net

LS-Net [**?**] was proposed as a single-shot line-segment detector inspired by SSD [**?**] and YOLO [**?**].

LS-Net reduces the need for pixel-level annotated data ground truth, which can be laborious to obtain. Especially in the case of powerlines, which can often be quite slender yet span across images, more meticulousness is required in the annotating process. Instead, LS-Net relies only on polyline annotation that trace along the powerline. Polyline annotation is much less effort and arguably more robust to labelling inaccuracy allowing more ground truth to be acquired facilitate better deep learning models.

LS-Net breaks down the problem of powerline detection into detecting and locating line segments in four overlapping grids, which are superimposed onto the input image. In each cell of the grids, LS-Net detect whether there exists parts of powerlines within its borders and provides the coordinates of the line segment endpoints. Theses divided results can be combined to produce a complete segment map of powerlines. The four-grid approach was proposed as opposed to one-grid approach, such as SSD [**?**] and YOLO [**?**], in order to encourage more thorough detection and localization of powerlines and combat the problem of discontinuities at grid cell borders and corners. This is illustrated in **??**.

To perform line segment detection, the architecture of LS-Net involves a fully convolutional feature extractor which branches out into a classifier module and a regressor module as shown in **??**. The modules is to detect line segment existence and line segment endpoint coordinates respectively. The feature extractor of the original LS-Net was proposed to be a version of VGG-16 network [**?**] which was modified to include Group Normalization [**?**] before activations and replace max pooling layer with stride 2 for convolutional layer. The classifier module uses the feature map from the extractor and follows up with a 2 x 2 convolutional layer with stride 1 resulting in four sets of 512-channeled feature maps, in which each element in the feature map correspond to a cell of the respective grid. Finally, a 1 x 1 convolutional layer reduces the feature maps to a 2-channeled output. The channels correspond to the classes ”line segment exists” and ”line segment does not exist”. The regressor module follows almost the identical design but ultimately produces a 4-channeled output instead, corresponding to the xy-coordinates of the predicted line segment endpoints. A detailed configuration of LS-Net is shown in **??**.

Loss ???

B. LS-Netv2

M: Instead of mentioning what we propose, we need to motivate it. Start from the problem (overlapping power lines) and use it to motivate the solution. Have a figure illustrating the whole pipeline and briefly describe the high-level method.

M: Then dive into the individual components (subsections), again starting from a motivation perspective

It has been shown that LS-Net has been successful for cases included in the dataset [], which consists mostly images of powerlines being captured from a relatively close distance and the visibility of the powerlines throughout the span of the image. These cases might resembles distance of footage from autonomous line following drones flying in proximity with the powerline. However, in reality, images such as those used for visual inspection can be taken from further away making the some segments of powerlines appear really thin and/or blended with the background. If we follow the grid-based approach of LS-Net, the detection performance of powerline in each cell may depend on the information of areas around it. However, according to this calculation the theoretic receptive field of LS-Net is only four times the cell size (? as calculated by torchscan [**?**]), which LS-Net is unable to aggregate global context for the cell inference. Furthermore, the design of the first LS-Net is based on the assumption that only one power line segment is present for each division cell and thus only

one segment is to be detected. This does not apply in practice where there exist powerlines appearing in close proximity and appearing to intersect each other (Figure **??**).

To alleviate these limitations of LS-Net, we propose a new version of LS-Net, called LS-Netv2. In order to tackle the thin powerline problem, the first change we made is to increase the input size from 512 to 1536 (3x). This is to cater to the visual inspection pipeline of powerlines which has to often deal with high resolution images from far away. We discovered that resizing these high resolution images to 1536 is the sweet spot where the trade-off between powerline visibility and model efficiency. Using the same compression ratio, the size of each grid cell becomes 96.

As mentioned, the VGG-inspired backbone design results in a receptive field of about four times the cell size. Furthermore, VGG architecture family has been quite obsolete and . Thus, we updated the backbone with a variant of the state-of-the-art ConvNeXt [**?**]. ConvNeXt was recently introduced as a modernized version of ResNet [**?**] which is equipped with recent properties of the hierarchical vision transformer Swin [**?**]. Modifications were made on the macro scale, such as the change in the ratios between stages, the use of depthwise convolution in similar manner to ResNeXt [**?**], enlarging kernel size to 7, etc.; and on the micro scale, such as the switch from batch normalization [**?**] to layer normalization [**?**], from ReLU [**?**] to GELU [**?**], etc. These changes allow ConvNeXt to compete favorably with other architectures, including Swin, which is among the state-of-the-art for deep vision tasks.

We use an altered version of ConvNeXt-Tiny, which is most compact and efficient ConvNeXt type. ConvNeXt-Tiny starts with the stem block that reduce the input image to a feature map 1/4th the original size. The stem block is then followed by has four downsampling stages, each of which ends with a compression convolutional layer with reduction factor of 2. Thus we use a ConvNeXt-Tiny, which is truncted at before the compression layer of the 3rd stage, as the new backbone. The output of the truncated model reduces would have the size of 96x96x384. The modified ConvNeXt model is concatenated with two additional convolutional layers to reduce the feature maps to size 31x31x256. With this design, the receptive field is increased to 1240, which is almost the size of the input (1536).

We take a step further by adding three vision transformer encoder blocks [**?**], [**?**] to allow for global aggregation of information. Originally, the input of transformer encoder are flattened patches of images, which are added with positional embeddings to convey location information of each patch to the model. In LS-Netv2, the image patches are replaced with 1x1x256 feature segments from the 31x31x256 feature maps. This design is similar to the object detector DETR [**?**], where a CNN is used to produce a compact feature map first then a transformer is used in the later stage. The CNN provide inductive bias that allow the model to be trained easier with smaller data. A pure transformer model would have more generalizability [**?**] but would rely on huge amount of data, which we do not have in our case, to achieve high performance. The benefit of this design will be shown using ablation study.

LS-Netv2 has the capability to detect multiple lines per cell by performing fixed number of inferences N, which is chosen to be larger than the maximum number of line segments exists in the cell. From our experiments, 10 is a safe choice. The ground truth y of each cell, which contains m actual line segments, is also perceived to contain N line segments $y = \{y_i\}_{i-1}^{N}$. However, the ground truth is now padded with $N - m$ negative classification. Inspired by DETR [**?**], we line segment detection within each cell as a direct set prediction problem. In our loss computation step, we include an optimal bipartite matching between the N predictions an N ground truths.

Let $\hat{y} = \{\hat{y}_i\}_{i-1}^{N}$ be the set of N predictions. The optimal bipartite matching between the N predictions an N ground truths, which is done using Hungarian algorithm, produces a permutation $\hat{\sigma} \in \mathfrak{S}$ that satisfies:

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}} \sum_{i}^{N} \mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)}) \tag{1}$$

where $\mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)})$ is the loss when pairing y_i and $\hat{y}_{\sigma(i)}$, which is reordered via permutation $\sigma(i)$. The loss considers both the classification loss and regression loss. Similarly to [**?**], this loss is defined as $-\hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_i=\text{positive}\}}\mathcal{L}_{reg}(b_i, \hat{b}_{\sigma(i)})$, where b_i and $\hat{b}_{\sigma(i)}$ are line segment endpoint ground truth and permutated prediction at index i respectively. $\mathcal{L}_{reg}(.)$ is the regression loss and $\hat{p}_{\sigma(i)}(c_i)$ is the predicted probability of class c_i, which is the classification ground truth at index i.

We propose a new version of LS-Net, called LS-Netv2. This version is updated with recent advancement in the deep learning space. Specifically, the feature extractor is altered with the state-of-the-art ConvNext model. Furthermore, inspired by DETR [**?**]. LS-Netv2 possess the ability to detect and locate multiple line-segments within each grid cell via multi-guess bipartite matching. Also, we propose the addition of a context module built from multi-head attentions layers, whose effectiveness in extracting global information has been proven. Finally, adjustments are made to the losses so that the training is more stable and requires less time to achieve plateau performance.

IV. Experiments

The procedure and results of experiments are presented. Comparisons to state-of-the-art are also conducted.

A. Datasets

1) Mendeley Powerline Dataset: The Mendeley Powerline dataset comprises of 400 images of size 512x512, among which 200 contain powerlines while the remaining only contain background. This dataset possesses pixel level annotation. To adapt the annotation to LS-Netv2, pixel annotation of individual powerlines are seperated via connected component analysis. Then, a skeletonization algorithm, introduced at [**?**], is used to produce polylines tracing the powerlines. The polylines are simplified using RamerDouglasPeucker algorithm [**?**]

and imposed on the 31x31 grid to produce the annotation for LS-Netv2.

skeleton-tracing [**?**], RamerDouglasPeucker algorithm [**?**]

2) PLD-UAV: PLD-UAV [**?**] contains two datasets of powerlines: power line dataset of urban scene (PLDU) and power line dataset of mountain scene (PLDM). In these datasets, the background are urban and mountain scene are quite cluttered and complex, however, the powerlines to be detected are still relatively observable. In this dataset, the boundary of powerlines are annotated at the pixel level. To adapt to LS-Netv2, we detected individual boundaries by dilating the pixel annotations and clustering the pixels via connected component analysis. From each cluster of pixels, a polygon is approximated and filled up with white pixels (positive labels). Then, skeletonization is used to produce the powerline tracing polylines, which can can be imposed onto the 31x31 grid to produce the annotation for LS-Netv2. Manual inspection was done to make sure the procedure produce accurate labels. PLDU contains 453 training data points and 120 testing data points. PLDM contains 237 training data points and 50 testing data points.

3) TTPLA: TTPLA [**?**] is a newly introduced dataset. It consists of images taken from UAVs under varieties of conditions, such as scenes, angles, zooming levels and lighting conditions. Many instances in this dataset are imposed to problems like occlusion and blending, which make the detection more challenging. The annotation of TTPLA is at instance segmentation level via polygons precisely wrapping around power lines. TTPLA also possesses many data points that have powerlines being close to each other andd powerline crossings, which will prove the effectiveness of the new multi-line-segment capability of LS-Netv2. To adapt it to LS-Netv2, similarly to the datasets above, we performed a image processing procedures which includes skeletonizing the blobs that fills the polygons annotations to generate polylines, which are simplified using RDP algorthim and then imposed on the 31x31 overlapping grid to find intersections, which are the labels for LS-Netv2. This dataset contains 1242 images, we uses 992 for training and 250 for testing. Examples are shown in Fig. **??**.

4) Proprietary dataset: We also evaluated the method using a dataset of powerlines aggregated by EsmartSystems. This dataset is made from UAVs images taken by multiple clients of EsmartSystems, and thus, contains significant diversity in terms of scenes, angles, zooming levels, weather and lighting conditions. This dataset has polyline annotations tracing the trajectories of the powerlines. The polyline annotation can be imposed onto the 31x31 grid to produce the annotation for LS-Netv2. In this dataset, the lines annotated includes conductors, guy wires and overhead ground wires. Conductors are normally in parallel. However, the other two type of wires can have different directions and, hence, there are many visual crossings. Included in the dataset are also some instances of line segments being in close proximity. The dataset contains 2961 images for training and 468 images for testing. Examples are shown in Fig. **??**.

B. Implementation Details

The proposed LSNetv2 is implemented with Tensorflow. Each LSNetv2 model is trained on one NVIDIA RTX 3090 24GB. The model was initialized using Xavier initialization [**?**] and trained with a batch size of 8 for 300 epochs. The Adam optimizer is used with learning rate of 0.0001, first momentum of 0.9 and second momentum of 0.99. The input image size is 512 x 512. During training, augmentation techniques applied include randomized occurences of sharpening, blur, color jittering, pixel dropouts and noises. After that randomized squared crops of varying size between 360 and 512 are taken from the input and then are scaled back to the input size of 512 x 512.

C. Evaluation Methods

For comparison purpose, we evaluate LSNetv2 in same manner as segmentation models. Similarly to [**?**], we adopt pixel-level averaged recall rate (ARR), averaged precision rate (APR) and F_1 Scores. In addition, we also use averaged F_β:

$$F_\beta = \frac{1}{N}\sum_{i=1}^{N}\frac{(1+\beta^2)\text{Precision}\times\text{Recall}}{\beta_2\times\text{Precision}+\text{Recall}} \tag{2}$$

where N is the number of test images and $\beta_2 = 0.3$ in order to place more emphasis on precision then conventional F_1. According to [**?**], recall is not as relevant as precision since the recall rate of 1.0 can be achieved by predicting all pixel in the image as positive. Also, in the case of LSNet, higher recall rate can also be attributed to imprecise prediction of line segment endpoints as illustrated in Fig **??**. Thus F_β score might a better measurement of LSNet performance than F_1. We also confirm the effectiveness of LSNetv2 via precision, recall and F_1 score of the classification branch as well as the regression loss of the regression branch.

To produce the segmentation map, for each cell within the 31x31 output grid that was classified to contain segments of powerlines, we use OpenCV to generate visible white lines from the predicted pairs of endpoints. Specific line widths are chosen for each dataset to reach the highest F_β scores. Particularly, we use the width of 9 pixels for PLDU dataset, 6 for PDLM dataset, 2 for TTPLA dataset and 5 for the proprierty dataset.

The comparison is done with the previously proposed LSNet across the four datasets. In addition, we also examined other versions of LSNetv2 where the backbone are varied to encourage the use of ConvNext. In particular, we looked at the original backbone of LSNet, a truncated Resnet-50 [**?**] and a truncated Efficientnetv2-M[**?**]. The truncation is performed so that the outputs of the backbones have the size of 32×32 from the input size of 512×512. The detailed architecture are shown in Table **??**. Specifically for TTPLA dataset, we also view LSNetv2 together with the reported performance results from PLGAN paper [**?**], which proposes PLGAN as the state-of-the-art for TTPLA dataset, to verify the competitiveness of LSNetv2. The performance results we looked at are of PLGAN itself, three segmentation models: FPN [**?**], DeepLabv3+ [**?**] and UNET++ [**?**]; ass well three line segment detectors: AFM

[**?**], LCNN [**?**] and HAWP [**?**]. Finally, ablation study is done to validate the newly proposed components.

D. Results

E. Ablation Study

V. Conclusion

The conclusion goes here.

Appendix A
Proof of the First Zonklar Equation

Appendix one text goes here.

Appendix B

Appendix two text goes here.

Acknowledgment

The authors would like to thank... [**?**]

PLACE PHOTO HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.

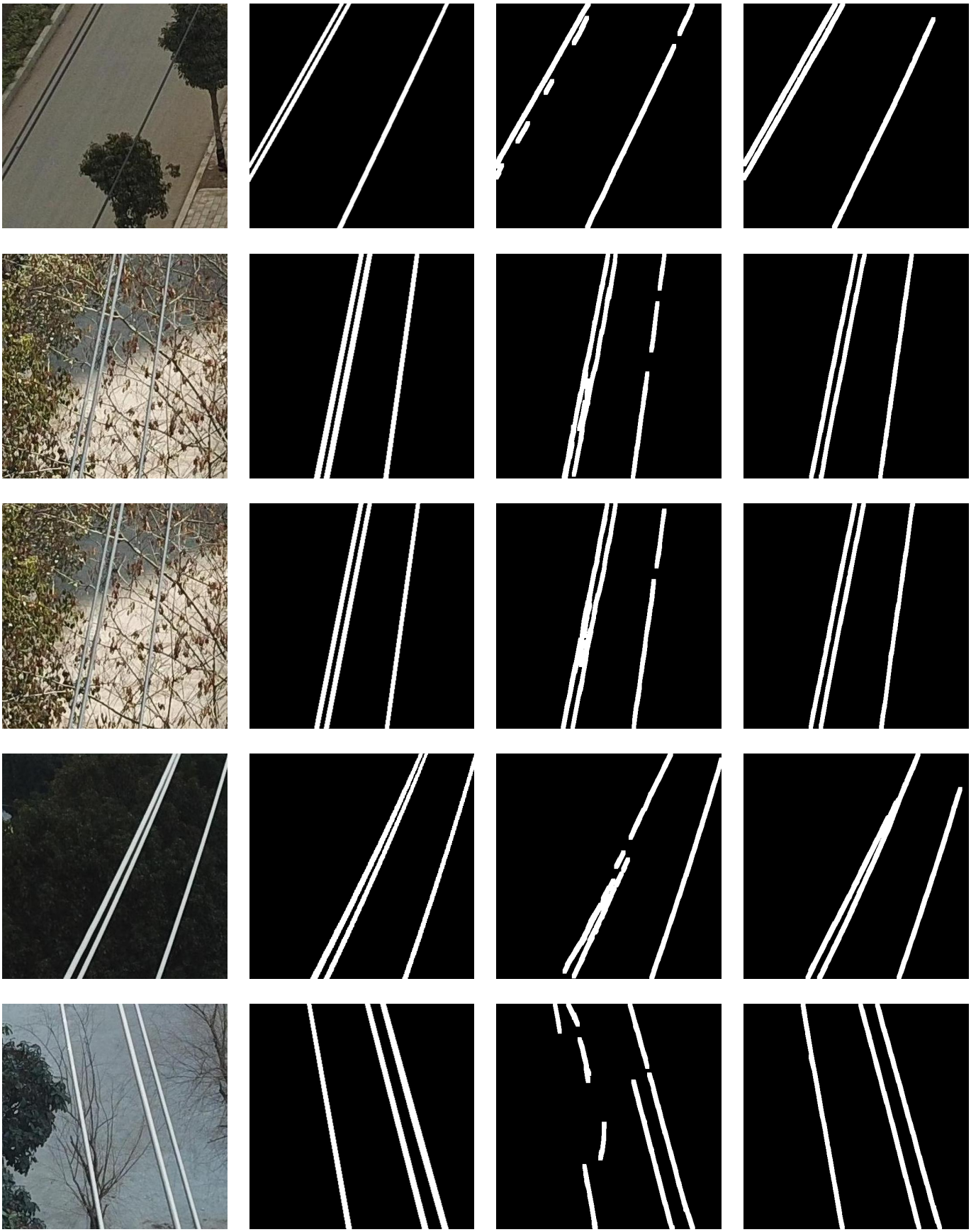

Fig. 1: 4 x 4

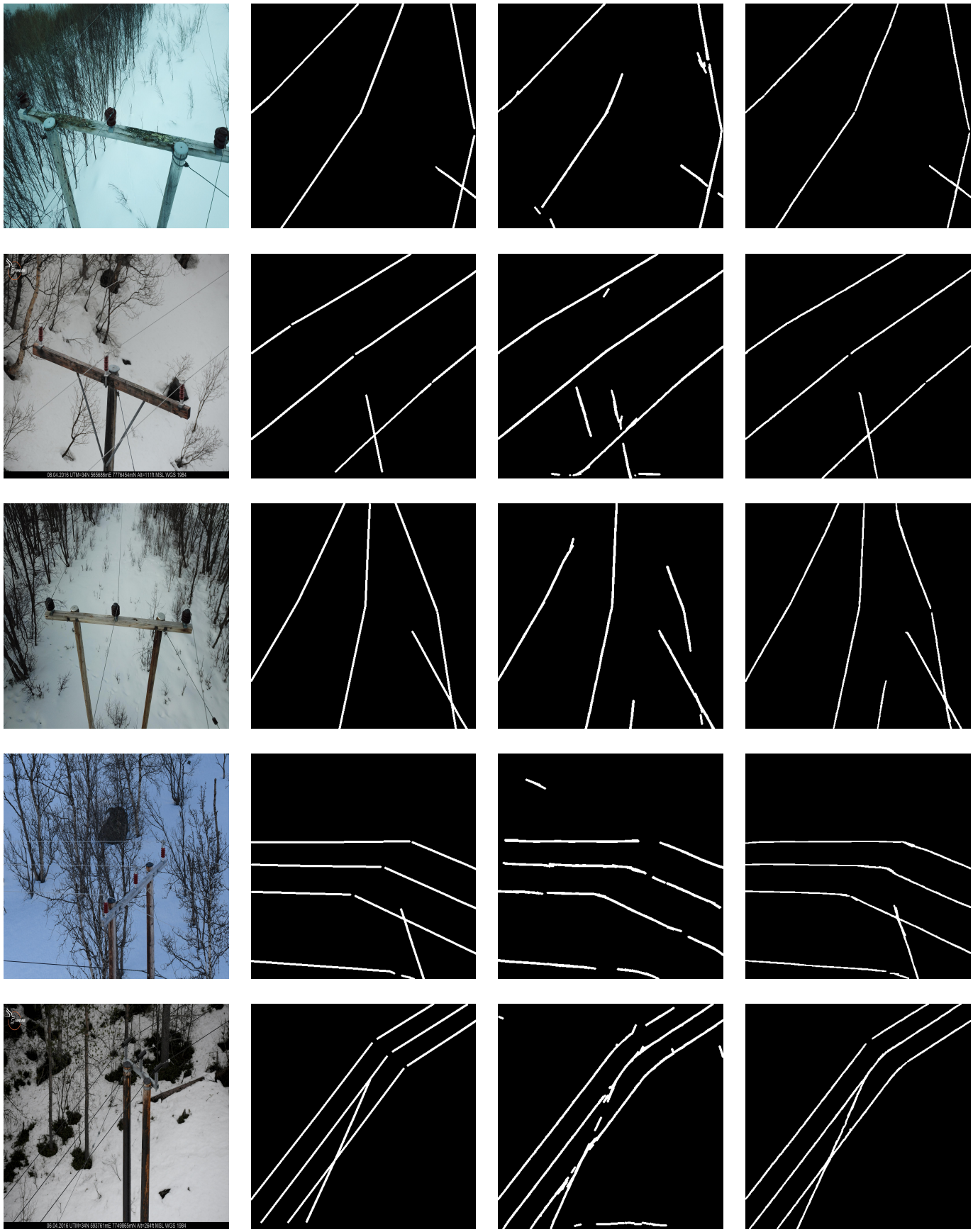

Image Ground truth LSNet LSNetv2

Fig. 2: 4 x 4

Image Ground truth LSNet LSNetv2

Fig. 3: 4 x 4

	PLDU				PLDM				TTPLA				Esmart			
	APR	ARR	F_1	F_β	APR	ARR	F_1	F_β	APR	ARR	F_1	F_β	APR	ARR	F_1	F_β
LSNet	0.914	0.662	0.767	0.840	0.916	0.726	0.810	0.864	0.579	0.525	0.551	0.565	0.726	0.812	0.766	0.744
LSNetv2	0.938	0.666	0.779	0.857	0.934	0.724	0.815	0.875	0.714	0.560	0.628	0.672	0.845	0.814	0.829	0.837

TABLE I: Result 1

	PLDU				PLDM				TTPLA				Esmart			
Backbone	APR	ARR	F_1	F_β	APR	ARR	F_1	F_β	APR	ARR	F_1	F_β	APR	ARR	F_1	F_β
modified VGG	0.899	0.685	0.778	0.839	0.818	0.793	0.805	0.812	0.696	0.566	0.624	0.661	0.759	0.813	0.785	0.771
Resnet-50	0.920	0.673	0.778	0.848	0.918	0.731	0.814	0.867	0.643	0.635	0.639	0.641	0.794	0.809	0.802	0.798
Efficientnetv2-M	0.907	0.645	0.754	0.829	0.923	0.674	0.779	0.851	0.573	0.355	0.438	0.502	0.747	0.796	0.771	0.758
ConvNext-Tiny	0.938	0.666	0.779	0.857	0.934	0.724	0.815	0.875	0.714	0.560	0.628	0.672	0.845	0.814	0.829	0.837

TABLE II: Backbone analysis

Image Ground truth LSNet LSNetv2

Fig. 4: 4 x 4

	Backbone	APR	ARR	F_1	F_β	param (M)
FPN*	Resnet-18	0.746	0.492	0.546	0.612	13.0
DeepLabv3+*	Resnet-18	0.784	0.510	0.573	0.645	12.3
UNET++*	Resnet-34	0.843	0.591	0.668	0.739	15.9
PLGAN*	Resnet-6	0.863	0.577	0.687	0.769	14.9
AFM	UNET	0.495	0.432	0.457	0.498	44.0
LCNN	Hourglass	0.541	0.464	0.315	0.498	10.9
HAWP	Hourglass	0.581	0.421	0.485	0.532	11.6
LSNetv2	modified VGG	0.696	0.566	0.624	0.661	10.0
LSNetv2	Resnet-50	0.643	0.635	0.639	0.641	12.8
LSNetv2	Efficientnetv2-M	0.573	0.355	0.438	0.502	14.5
LSNetv2	ConvNext-Tiny	0.714	0.560	0.628	0.672	13.9

TABLE III: Comparison to TTPLA results.

ConvNext	Bipartite	Ordered loss	APR	ARR	F_1	F_β
			0.726	0.812	0.766	0.744
✓			0.844	0.770	0.806	0.826
✓	✓		0.825	0.820	0.823	0.824
	✓		0.743	0.787	0.764	0.753
	✓	✓	0.759	0.813	0.785	0.771
✓	✓	✓	0.845	0.814	0.829	0.837

TABLE IV: Esmart ablation.

ConvNext	Bipartite	Ordered loss	APR	ARR	F_1	F_β
			0.579	0.525	0.551	0.565
✓			0.624	0.660	0.641	0.632
✓	✓		0.651	0.664	0.616	0.650
	✓		0.671	0.550	0.604	0.638
	✓	✓	0.696	0.566	0.624	0.661
✓	✓	✓	0.714	0.560	0.628	0.672

TABLE V: TTPLA ablation.

Stage	Output size	Modified VGG	Resnet-50	EfficientnetV2-M	ConvNext-Tiny
1	256 x 256			$3\times3, 24$, s=2 $3\times \begin{cases} 3\times 3, 24 \end{cases}$	
2	128 x 128		1×1 maxpool, s=2 $2\times \begin{cases} 1\times 1, 64 \\ 3\times 3, 64 \\ 1\times 3, 64 \end{cases}$	$3\times 3, 96$, s=2 $1\times 1, 48$ $4\times \begin{cases} 3\times 3, 192 \\ 1\times 1, 48 \end{cases}$	$4\times4, 96$, s=4
3	64 x 64			$3\times 3, 192$, s=2 $1\times 1, 80$ $4\times \begin{cases} 3\times 3, 320 \\ 1\times 1, 80 \end{cases}$	$3\times \begin{cases} 7\times 7, 96 \\ 1\times 1, 384 \\ 1\times 1, 96 \end{cases}$ $2\times2, 192$, s=2
4	32 x 32			$1\times 1, 320$ dw $3\times 3, 320, s=2$ SE-Block(ratio=0.25) $1\times 1, 160$ $6\times \begin{cases} 1\times 1, 640 \\ \text{dw } 3\times 3, 640 \\ \text{SE-Block(ratio=0.25)} \\ 1\times 1, 160 \end{cases}$	$3\times \begin{cases} \text{dw } 7\times 7, 192 \\ 1\times 1, 768 \\ 1\times 1, 192 \end{cases}$ $2\times2, 384$, s=2
5	32 x 32			$1\times 1, 960$ dw $3\times 3, 960$ SE-Block(ratio=0.25) $1\times 1, 176$ $13\times \begin{cases} 1\times 1, 1056 \\ \text{dw } 3\times 3, 1056 \\ \text{SE-Block(ratio=0.25)} \\ 1\times 1, 176 \end{cases}$ $1\times 1, 1056$	$9\times \begin{cases} \text{dw } 7\times 7, 384 \\ 1\times 1, 1536 \\ 1\times 1, 384 \end{cases}$
6					
7					

TABLE VI: Backbone architecture.