Occupancy and Flow Prediction Using Temporal Convolutions

[1]Daeil Han, [1]Gaeun Kim, [2]Yeong Jun Koh, [1]Hanul Kim
[1]Seoul National University of Science and Technology
[2]Chungnam National University
{dalehan0606, gekim}@seoultech.ac.kr, yjkoh@cnu.ac.kr, hukim@seoultech.ac.kr

Abstract

Occupancy and flow prediction is essential for autonomous driving, as it aims to forecast the future positions and movements of various objects, such as vehicles and pedestrians. In this report, we present our approach to the occupancy and flow prediction task in the 2024 Waymo Open Dataset Challenge. Our model integrates a spatial-temporal encoder, a multi-scale aggregator, and an autoregressive decoder. We employ multiple loss functions, including focal loss, flow loss, and a modified traced loss. Our method achieves a Flow-Grounded AUC of 0.7564 on the test set of the 2024 Waymo Open Dataset.

1. Proposed Method

Figure 1 depicts our model, which includes four components: encoder, aggregator, decoder, and prediction head. The encoder first takes two types of input features, called agent and static features, and encodes them into a multi-scale feature map. The aggregator then improves this feature map. Using the enhanced feature map, the decoder autoregressively generates features for future frames. Lastly, the prediction head estimates the occupancy and flow maps for these future frames.

1.1. Input Representation

For a given scenario, we extract 10 previous frames along with one current frame. Each frame includes agent and static feature maps, which are bird's-eye-view (BEV) feature maps aligned to the ego vehicle. The agent feature $A_t \in \mathbb{R}^{H\times W\times C}$ captures the information of dynamic agents at frame t. Here, $H = 256$ and $W = 256$ denote the height and the width of a feature map, and $C = 3$ is the number of input feature channels. Each channel of the agent feature represents object type, x-direction velocity, and y-direction velocity. The object type can be one of three values: vehicle object (+1), non-vehicle object (-1), and absence of object (0). On the other hand, the static feature $S_t \in \mathbb{R}^{H\times W\times C}$ encodes the road environment, with its channels detailing road type, road angle, and traffic light states. We define our inputs by concatenating each feature along time dimension. Therefore, the inputs $\mathcal{A} = [A_1, \cdots, A_T]$ and $\mathcal{S} = [S_1, \cdots, S_T]$ are 4-dimensional tensors of size $T \times H \times W \times C$, where $T = 11$ denotes the index of the current frame.

1.2. Model

Encoder: Figure 2 illustrates the detailed architecture of our encoder. We base our encoder on the CAFormer-S18 [13] backbone, pretrained on the ImageNet dataset [11]. Thus, our encoder shares many hyperparameters with the CAFormer-S18, such as the number of stages, the number of blocks per stage, and the dimensions for each stage. However, since the CAFormer-S18 backbone is designed to classify static images, we modify it to handle temporal input.

Specifically, our encoder stage consists of four components: a spatial downsample layer, a metaformer block, a temporal convolution block, and a temporal downsample layer. The first two components are derived from CAFormer-S18. We set the stride of the first spatial downsample layer to 4, and the stride for the other spatial downsample layers to 2. The temporal convolution block performs layer normalization [1] and two 1D convolutions with a SiLU [10] activation function. The temporal downsample is a strided 1D convolution layer to decrease the temporal resolution of the inputs by half.

Given our inputs $\mathcal{A}$ and $\mathcal{S}$, the encoder first processes them using the spatial downsample layers of the first stage, and then combines them for subsequent blocks. We then perform global average pooling on the output of each stage along the temporal dimension to produce multi-scale features $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3, \mathcal{X}_4$. The ith scale feature map $\mathcal{X}_i \in \mathbb{R}^{H_i\times W_i\times C_i}$ is a 3-dimensional tensor, where H_i, W_i, and C_i are the height, width, and number of channels at this scale, respectively. Here, $H_i = H/2^{i+1}$ and $W_i = W/2^{i+1}$.

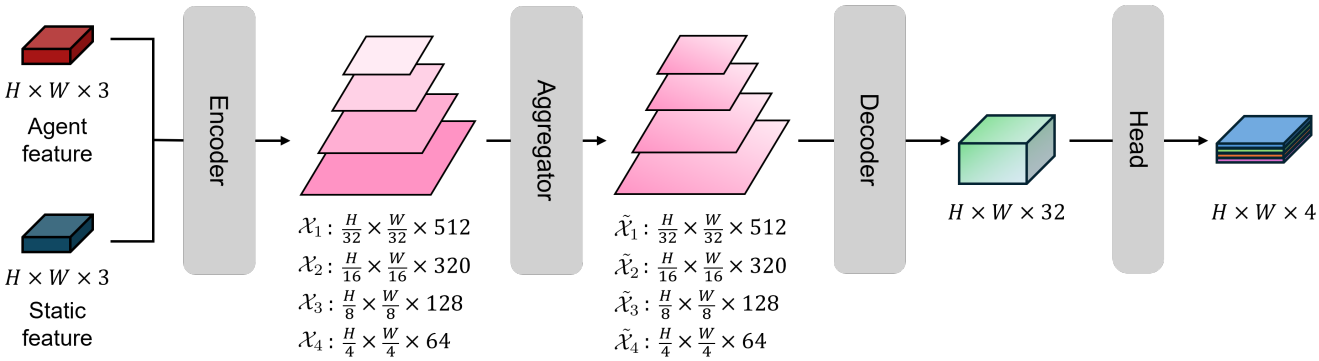


Figure 1. The overview of our model.

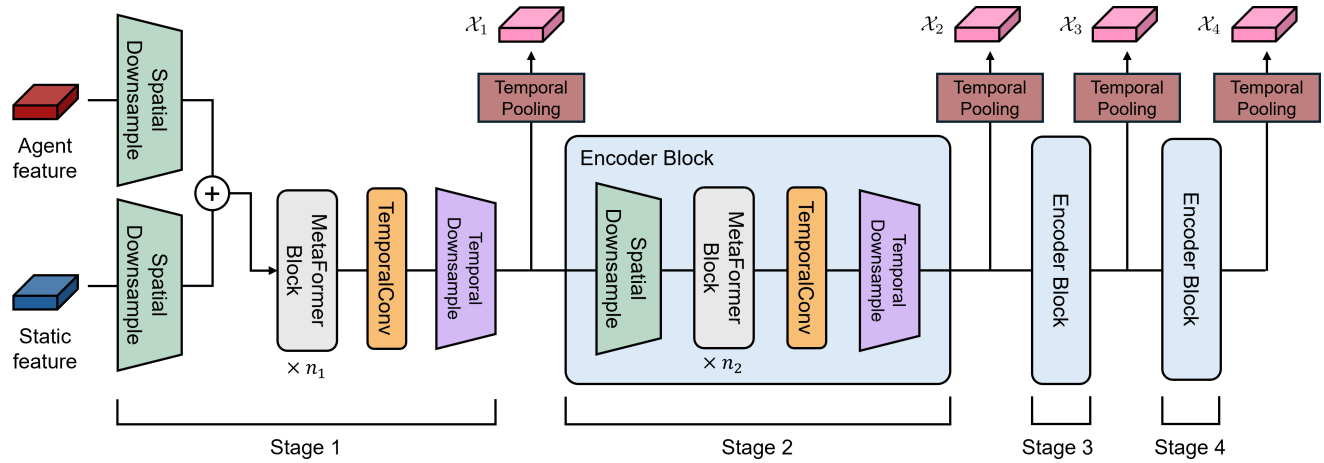


Figure 2. The structure of the encoder using MetaFormer with temporal convolutions.

Aggregator: The aggregator is designed to improve multi-scale features by incorporating precise localization information from fine-scale features and adding richer semantic context from coarse-scale features. To achieve this, we employ Path Aggregation Network (PAN) [5] in YOLO-v9 [12]. The output of the aggregator is the enhanced features at each scale. We denote the enhanced feature at the ith scale as $\tilde{\mathcal{X}}_i \in \mathbb{R}^{H_i \times W_i \times C_i}$.

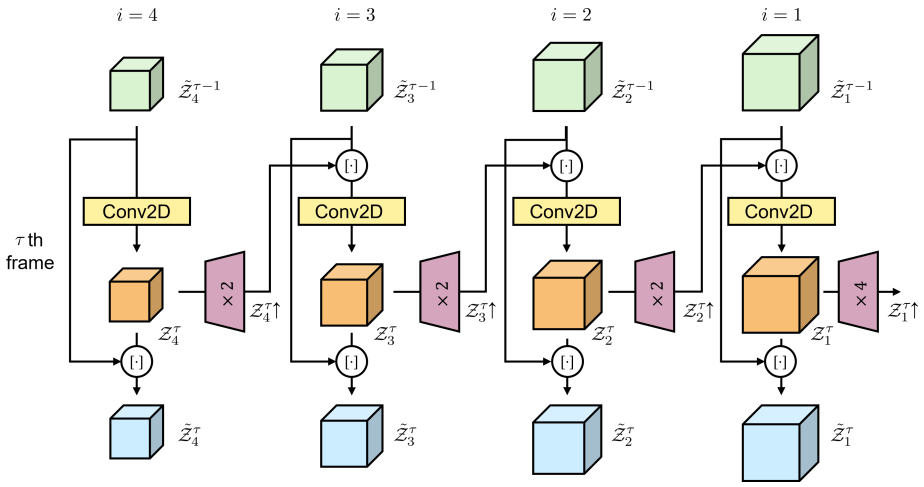
Decoder: The goal of the decoder is to predict feature maps for future frames. To achieve this, we design an autoregressive decoder based on convolution. As illustrated in Figure 3, our decoder consists of a decoder cell for each scale, each containing a convolution layer with SiLU activation and a concatenation operation. For clarity, let us consider the operation of the ith scale decoder cell for the τth frame prediction: This cell takes two inputs: the feature from the previous scale $\mathcal{Z}_{i+1}^{\tau} \uparrow$ and the feature from the previous time step $\tilde{\mathcal{Z}}_i^{\tau-1}$. Initially, the feature from the previous time step $\tilde{\mathcal{Z}}_i^{0}$ is set to the output of the aggregator $\tilde{\mathcal{X}}_i$. Given these inputs, the cell stacks them and performs convolution filtering. It then produces two output features: one for the upper scale cell $\mathcal{Z}_i^{\tau} \uparrow$ and one for the next time step $\tilde{\mathcal{Z}}_i^{\tau}$.

Let $\mathcal{Z}_i^{\tau}$ be the result of the convolution operation. For the upper scale cell, we define the output $\mathcal{Z}_i^{\tau} \uparrow$ as the upsampled version of $\mathcal{Z}_i^{\tau}$, using bilinear interpolation for upsampling. For the next time step, we define the output $\tilde{\mathcal{Z}}_i^{\tau}$ as

$$\tilde{\mathcal{Z}}_i^{\tau} = [\tilde{\mathcal{Z}}_i^{0}, \mathcal{Z}_i^{1}, \cdots, \mathcal{Z}_i^{\tau}] \tag{1}$$

where $[\cdot]$ represents the concatenation operation. This means $\tilde{\mathcal{Z}}_i^{\tau}$ retains all temporal information from the current frame up to the future τth frame. Therefore, the prediction for $\tau + 1$ is performed autoregressively.

Prediction Head: The upscaled outputs from the finest-scale decoder cell $\mathcal{Z}_1^{\tau} \uparrow \in \mathbb{R}^{H \times W \times C_d}$ are fed into the prediction head to estimate the occupancy flow map for the



Feature	Shape
$\mathcal{Z}_4^{\tau}$	$32 \times \frac{H}{32} \times \frac{W}{32}$
$\mathcal{Z}_3^{\tau}$	$32 \times \frac{H}{16} \times \frac{W}{16}$
$\mathcal{Z}_2^{\tau}$	$32 \times \frac{H}{8} \times \frac{W}{8}$
$\mathcal{Z}_1^{\tau}$	$32 \times \frac{H}{4} \times \frac{W}{4}$
$\mathcal{Z}_4^{\tau}\uparrow$	$32 \times \frac{H}{16} \times \frac{W}{16}$
$\mathcal{Z}_3^{\tau}\uparrow$	$32 \times \frac{H}{8} \times \frac{W}{8}$
$\mathcal{Z}_2^{\tau}\uparrow$	$32 \times \frac{H}{4} \times \frac{W}{4}$
$\mathcal{Z}_1^{\tau}\uparrow$	$32 \times H \times W$
$\tilde{\mathcal{Z}}_4^{\tau}$	$(256 + 32\tau) \times \frac{H}{32} \times \frac{W}{32}$
$\tilde{\mathcal{Z}}_3^{\tau}$	$(160 + 32\tau) \times \frac{H}{16} \times \frac{W}{16}$
$\tilde{\mathcal{Z}}_2^{\tau}$	$(64 + 32\tau) \times \frac{H}{8} \times \frac{W}{8}$
$\tilde{\mathcal{Z}}_1^{\tau}$	$(32 + 32\tau) \times \frac{H}{4} \times \frac{W}{4}$

Figure 3. The architecture of the decoder and the table of feature shapes used in our model at the τ frame.

future frame τ. Here, $C_d = 32$, which is the dimension of the decoder cell. The prediction head is designed with two convolution layers and SiLU activation to generate the occupancy and flow maps. The final output maps include observed occupancy $\hat{\mathcal{O}}_{\text{obs}}^{\tau} \in \mathbb{R}^{H\times W\times 1}$, occluded occupancy $\hat{\mathcal{O}}_{\text{occ}}^{\tau} \in \mathbb{R}^{H\times W\times 1}$, and flow $\hat{\mathcal{F}}^{\tau} \in \mathbb{R}^{H\times W\times 2}$. The flow map's two channels represent the flow in the x and y directions in the coordinate system centered on the ego vehicle.

1.3. Loss Function

We train our model by minimizing three loss terms: focal loss $\mathcal{L}_{\text{focal}}$, flow loss $\mathcal{L}_{\text{flow}}$, and traced loss $\mathcal{L}_{\text{traced}}$. The total loss function is defined as follows:

$$\mathcal{L} = \lambda_{\text{focal}}\mathcal{L}_{\text{focal}} + \lambda_{\text{flow}}\mathcal{L}_{\text{flow}} + \lambda_{\text{trace}}\mathcal{L}_{\text{trace}} \tag{2}$$

where λ_{focal}, λ_{flow}, and λ_{trace} are hyperparameters used to balance the contributions of each loss term. We empirically set these hyperparameters to $\lambda_{\text{focal}} = 5$, $\lambda_{\text{flow}} = 1$, and $\lambda_{\text{trace}} = 0.02$.

Focal Loss: Focal loss penalizes classification errors in predicted occupancy maps. Let $\mathcal{O}_{\text{obs}}^{\tau}$ and $\mathcal{O}_{\text{occ}}^{\tau}$ be the ground-truth occupancy maps at time step τ. We define the binary focal loss as

$$\mathcal{L}_{\text{obs}} = \texttt{focal}\left(\hat{\mathcal{O}}_{\text{obs}}^{\tau}, \mathcal{O}_{\text{obs}}^{\tau}\right) \tag{3}$$

$$\mathcal{L}_{\text{occ}} = \texttt{focal}\left(\hat{\mathcal{O}}_{\text{occ}}^{\tau}, \mathcal{O}_{\text{occ}}^{\tau}\right) \tag{4}$$

$$\mathcal{L}_{\text{focal}} = \mathcal{L}_{\text{obs}} + \mathcal{L}_{\text{occ}} \tag{5}$$

where $\texttt{focal}(\cdot,\cdot)$ denotes the standard binary focal loss [4], and $\mathcal{L}_{\text{obs}}$ and $\mathcal{L}_{\text{occ}}$ are the computed losses for observed and occluded occupancy maps, respectively.

Flow Loss: We use a masked L1-loss for our flow loss. This loss is defined as follows:

$$\mathcal{L}_{\text{flow}} = \left\| \mathcal{M}^{\tau} \odot \left(\hat{\mathcal{F}}^{\tau} - \mathcal{F}^{\tau}\right) \right\|_1 \tag{6}$$

where $\odot$ represents element-wise multiplication, and $\mathcal{M}^{\tau} \in \mathbb{R}^{H\times W\times 2}$ is a binary mask that indicates the presence of ground-truth flow. The mask $\mathcal{M}^{\tau}$ is calculated as:

$$\mathcal{M}^{\tau}(i,j) = \begin{cases} 1 & \text{if } \mathcal{F}^{\tau}(i,j) \neq \mathbf{0} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where $\mathbf{0} \in \mathbb{R}^2$ is a zero vector, and $\mathcal{M}^{\tau}(i,j)$ and $\mathcal{F}^{\tau}(i,j)$ denote the mask values and the ground-truth flow at the coordinates (i,j), respectively.

Traced Loss: Traced loss [8] encourages both accurate occupancy and flow predictions. Let $\mathcal{O}^{\tau-1} = \mathcal{O}_{\text{obs}}^{\tau-1} \cup \mathcal{O}_{\text{occ}}^{\tau-1}$ represent the union of the ground-truth observed and occluded occupancy maps. We warp this union to the frame τ using the predicted flow map $\hat{\mathcal{F}}^{\tau}$, resulting in the warped occupancy map $\mathcal{O}^{\tau-1\to\tau}$. The traced loss is then defined as

$$\mathcal{L}_{\text{trace}} = \texttt{focal}\left(\mathcal{O}^{\tau-1\to\tau}, \mathcal{O}^{\tau}\right) \tag{8}$$

Here, $\mathcal{O}^{\tau}$ is the union of the ground-truth observed and occluded occupancy maps at frame τ.

2. Experiments

2.1. Implementation Details

We evaluate our method using the Waymo Open Motion dataset [2]. For each scenario, we utilize 11 frames of past and present data and extract agent and static features from

Table 1. 2024 Waymo Open Dataset test set performance comparison. The best results are in bold.

	Observed		Occluded		Flow	Flow-Grounded	
Model	AUC ↑	Soft IoU ↑	AUC ↑	Soft IoU ↑	EPE ↓	AUC ↑	Soft IoU ↑
DOPP	**0.7972**	0.3429	**0.1937**	0.0241	**2.9574**	**0.8026**	0.5156
STrajNet	0.7514	0.4818	0.1610	0.0183	3.5867	0.7772	**0.5551**
VectorFlow	0.7548	**0.4884**	0.1736	**0.0448**	3.5827	0.7669	0.5298
HGNET	0.7332	0.4211	0.1656	0.0180	3.6699	0.7403	0.4498
Ours	0.7552	0.2299	0.1658	0.0180	3.3779	0.7564	0.4431

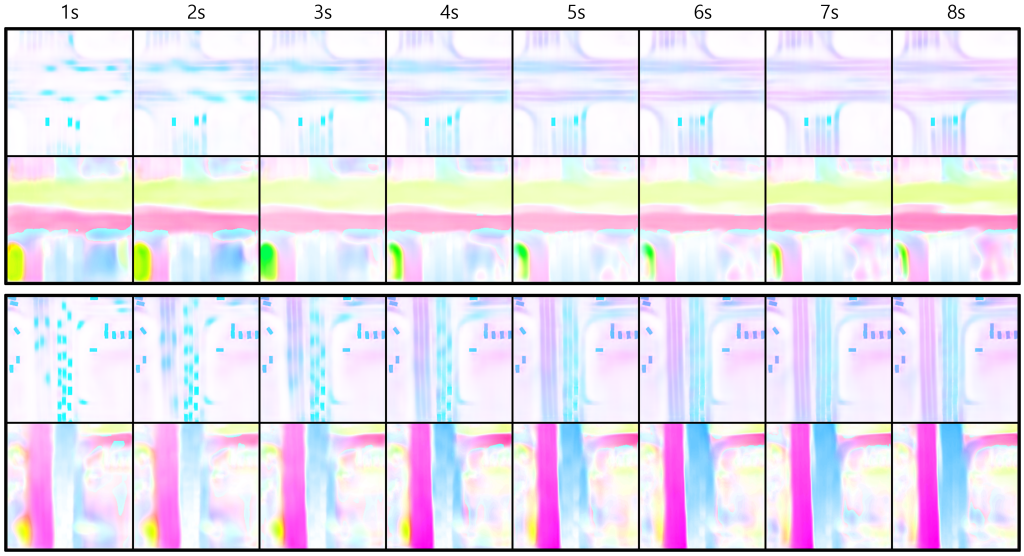


Figure 4. Visualization of predictions for two scenes.

each frame. These features are 256×256 bird-eye-view maps, representing an area of $80 \times 80\,\mathrm{m}^2$ in the real world. For training, we use the AdamW [7] optimizer with the initial learning rate of 2×10^{-4} and weight decay of 0.05. We train our model for 10 epochs with cosine annealing [6]. We set the batch size to 72.

2.2. Results

Table 1 compares our method with competing methods in the 2024 Open Waymo Challenge. In Table 1, our method ranks 4th in terms of the Flow-Grounded AUC, the primary performance metric in this challenge. This is likely due to our model being less trained (10 epochs) and not utilizing enriched input information beyond the 256×256 maps, which have been reported to be effective in previous studies [3, 9]. Therefore, improving these aspects remains our future work. Figure 4 visualizes the predicted occupancy and flow of our method for two scenes. Each column represents future predictions from 1 second to 8 seconds ahead. The top row shows the predicted occupancy, while the bottom row displays the predicted flow.

3. Conclusion

In this report, we introduced our solution for the occupancy and flow prediction task in the 2024 Waymo Open Dataset Challenge. We designed our model using a spatio-temporal encoder, a multi-scale aggregator, and an autoregressive decoder. For the encoder, we utilized CAFormer-S18 [13], expanding it to handle temporal features. To aggregate the encoder outputs, we employed the PAN structure from YOLOv9 [12]. Additionally, we developed a convolution-based decoder to generate future frames autoregressively. Our method achieves a Flow-Grounded AUC of 0.7564 on the test set. However, our method is limited by its relatively short training duration and the lack of enriched input information. Future work will focus on extending training and incorporating richer input data to enhance performance.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 1

[2] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi

Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles Qi, Yin Zhou, Zoey Yang, Aurelien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving : The waymo open motion dataset, 2021. 3

[3] Yihan Hu, Wenxin Shao, Bo Jiang, Jiajie Chen, Siqi Chai, Zhening Yang, Jingyu Qian, Helong Zhou, and Qiang Liu. Hope: Hierarchical spatial-temporal network for occupancy flow prediction, 2022. 4

[4] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018. 3

[5] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation, 2018. 2

[6] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017. 4

[7] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 4

[8] Reza Mahjourian, Jinkyu Kim, Yuning Chai, Mingxing Tan, Ben Sapp, and Dragomir Anguelov. Occupancy flow fields for motion forecasting in autonomous driving. *IEEE Robotics and Automation Letters*, 7(2):5639–5646, 2022. 3

[9] Dmytro Poplavskiy. Waymo open dataset occupancy and flow prediction challenge solution: Look around, 2022. Accessed: 2024-05-30. 4

[10] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. 1

[11] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015. 1

[12] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. Yolov9: Learning what you want to learn using programmable gradient information, 2024. 2, 4

[13] Weihao Yu, Chenyang Si, Pan Zhou, Mi Luo, Yichen Zhou, Jiashi Feng, Shuicheng Yan, and Xinchao Wang. Metaformer baselines for vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(2):896–912, 2024. 1, 4