Wind Farm Power Prediction with Graph Neural Network

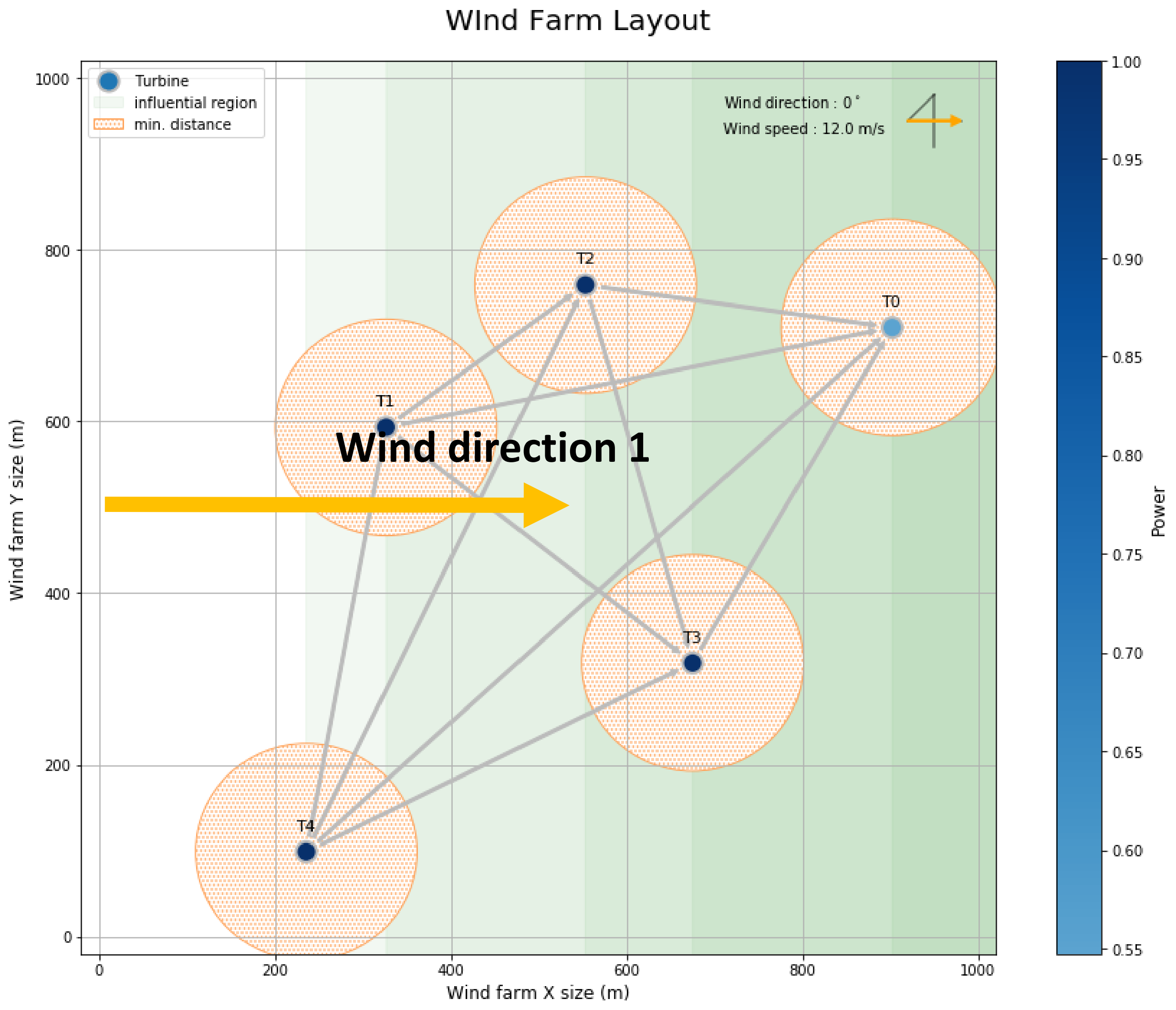
Junyoung Park

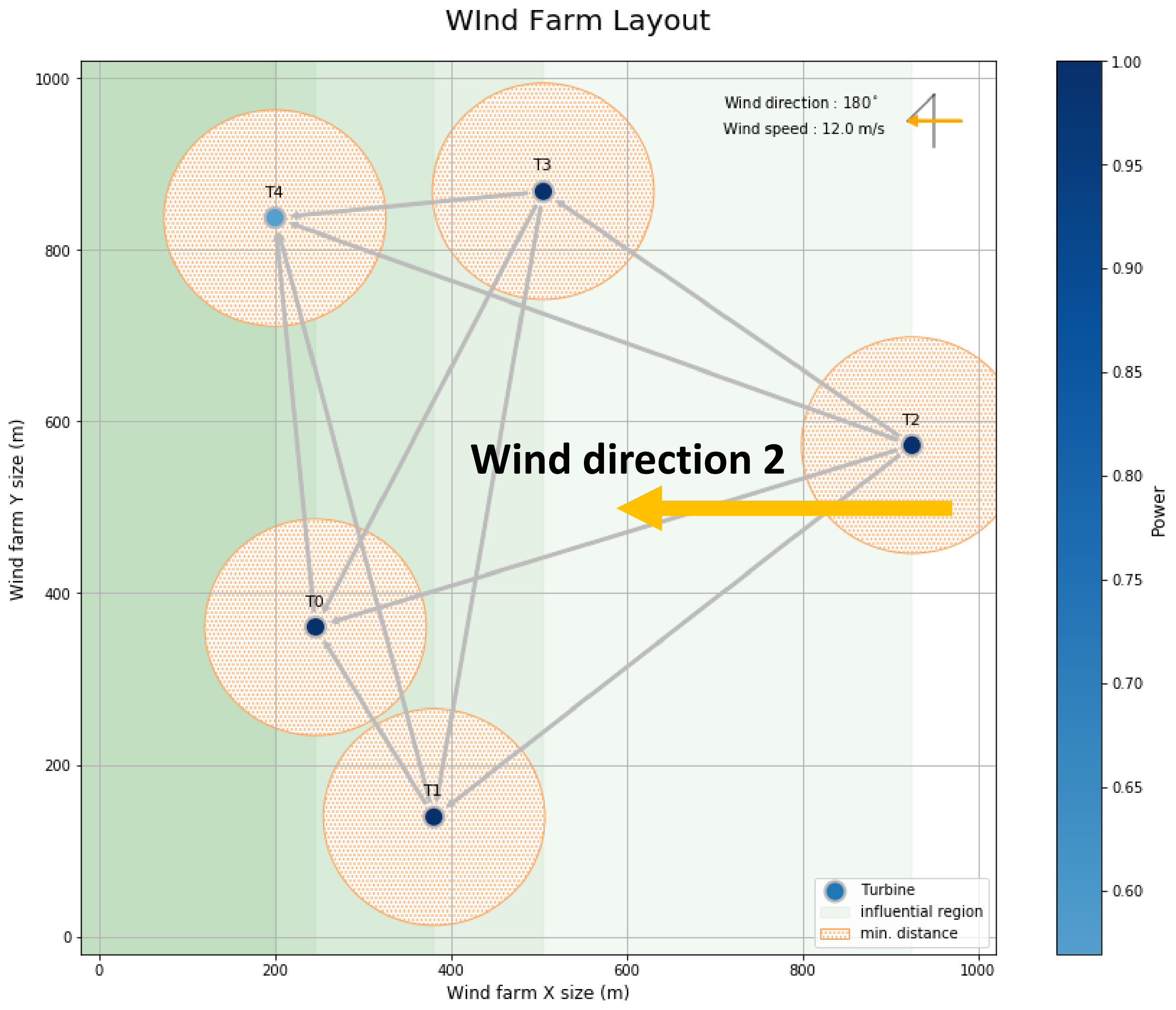
SYSTEMS INTELLIGENCE Lab

Industrial and Systems Engineering (ISysE)

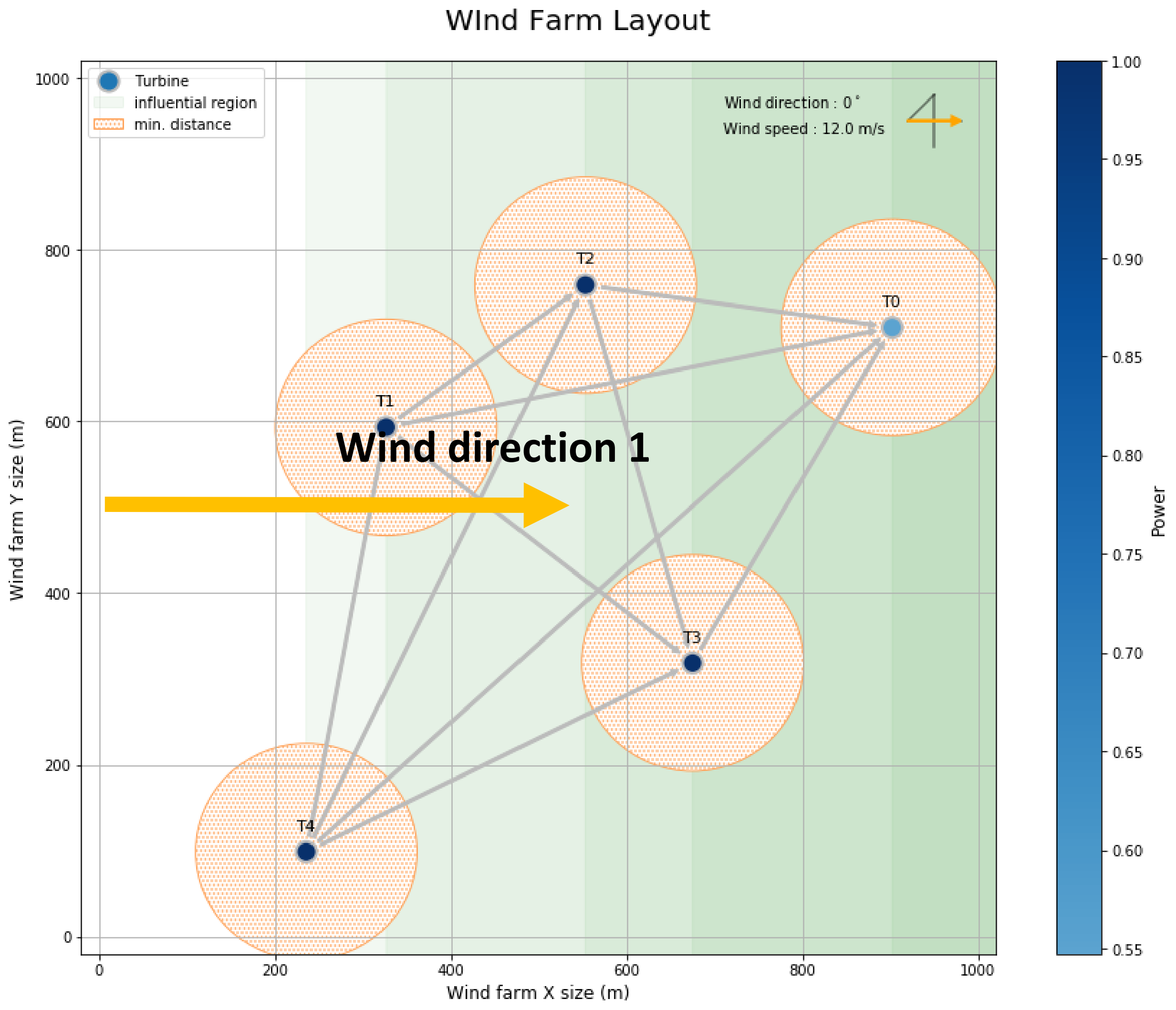
KAIST

Wind Farm Power Estimation Task





Wind Farm Power Estimation Task



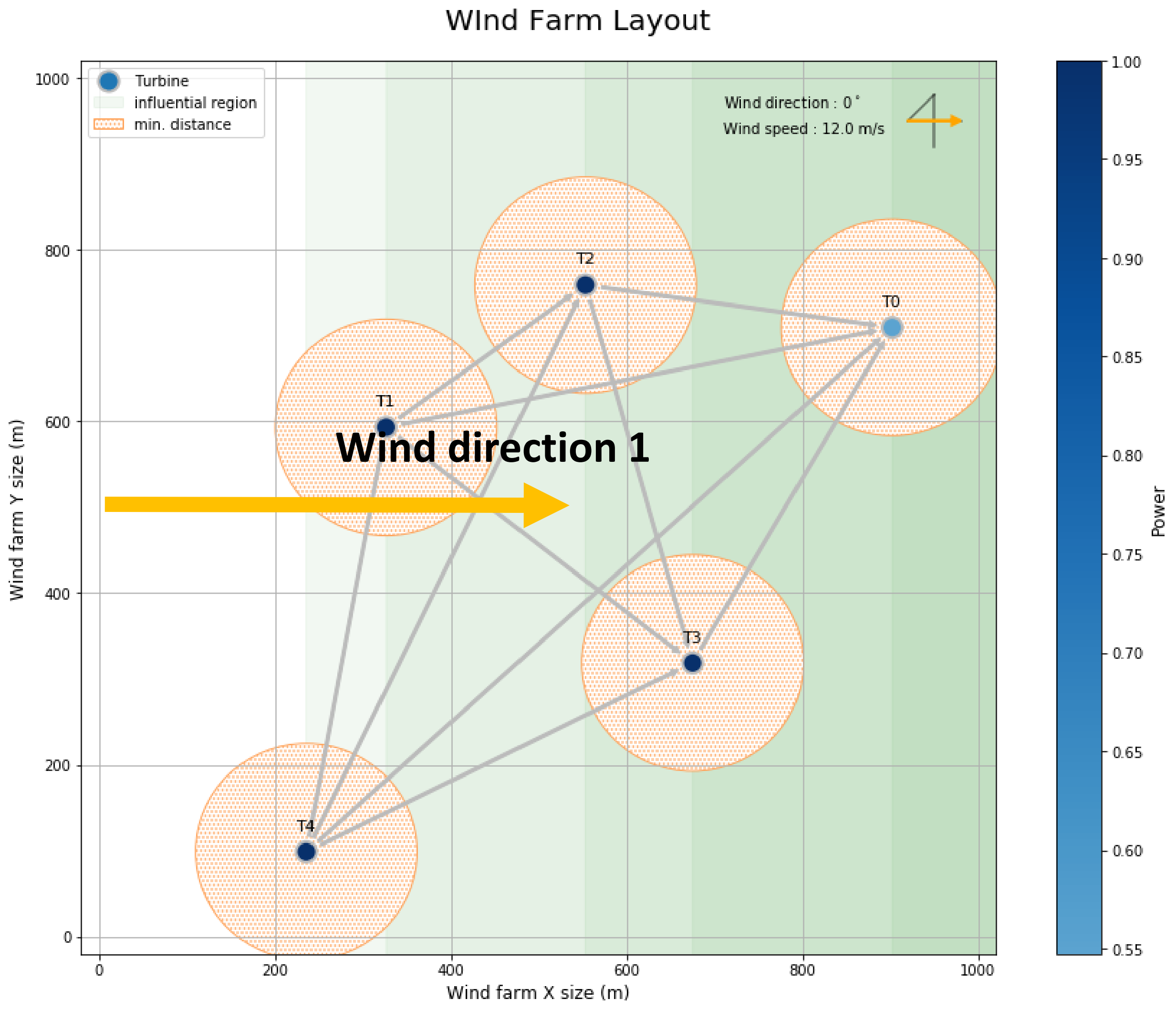
- Farm-level power estimation

  Wind-farm power = ??

- Turbine-level power estimation

  Wind turbine powers = ??

Wind Farm Power Estimation Task



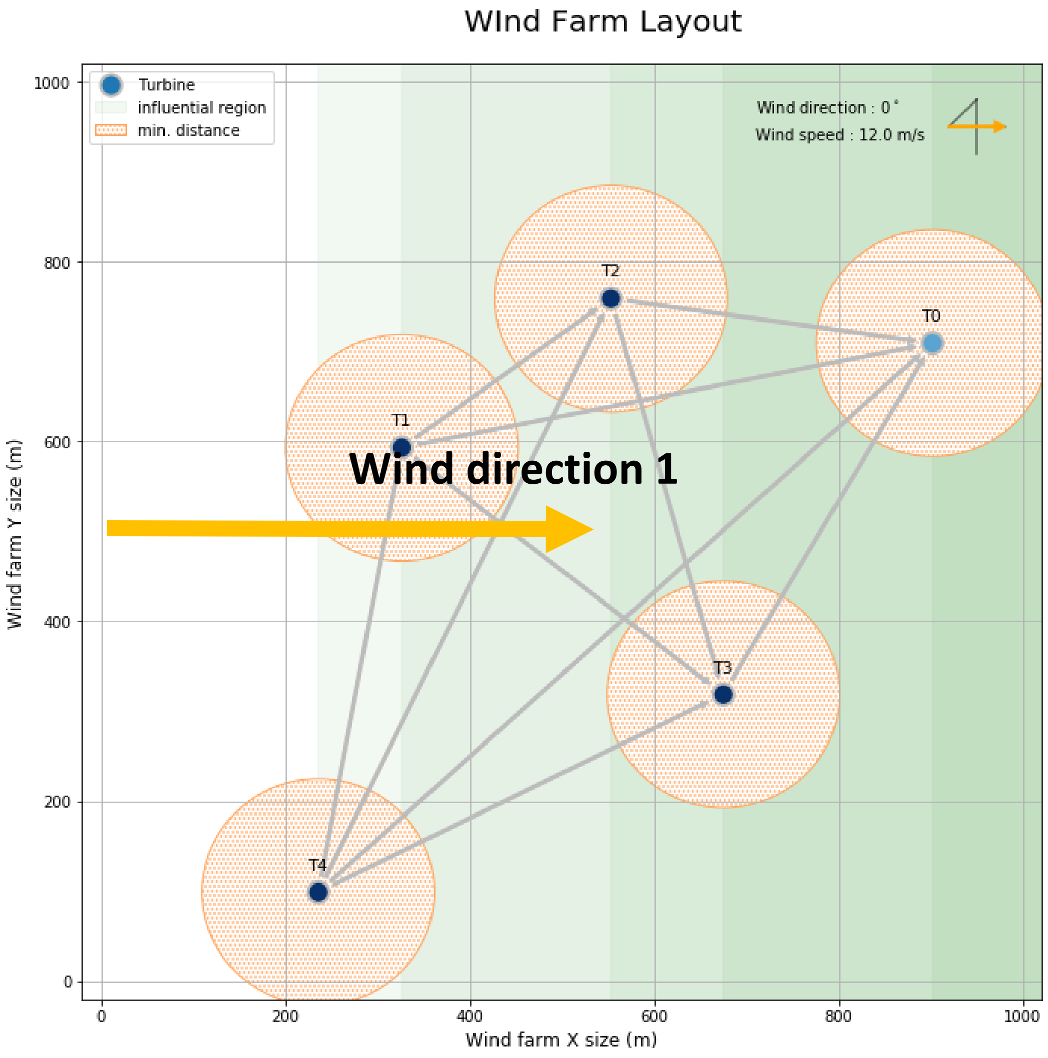
- Farm-level power estimation

  Wind-farm power = ??

- Turbine-level power estimation

  Wind turbine powers = ??

Wind Farm and Its Graph Representation



$$\mathcal{G} = (N, E, g)$$

Node features N = $\{free\ flow\ wind\ speed\}_{\forall i\,\in\, turbine\ index}$

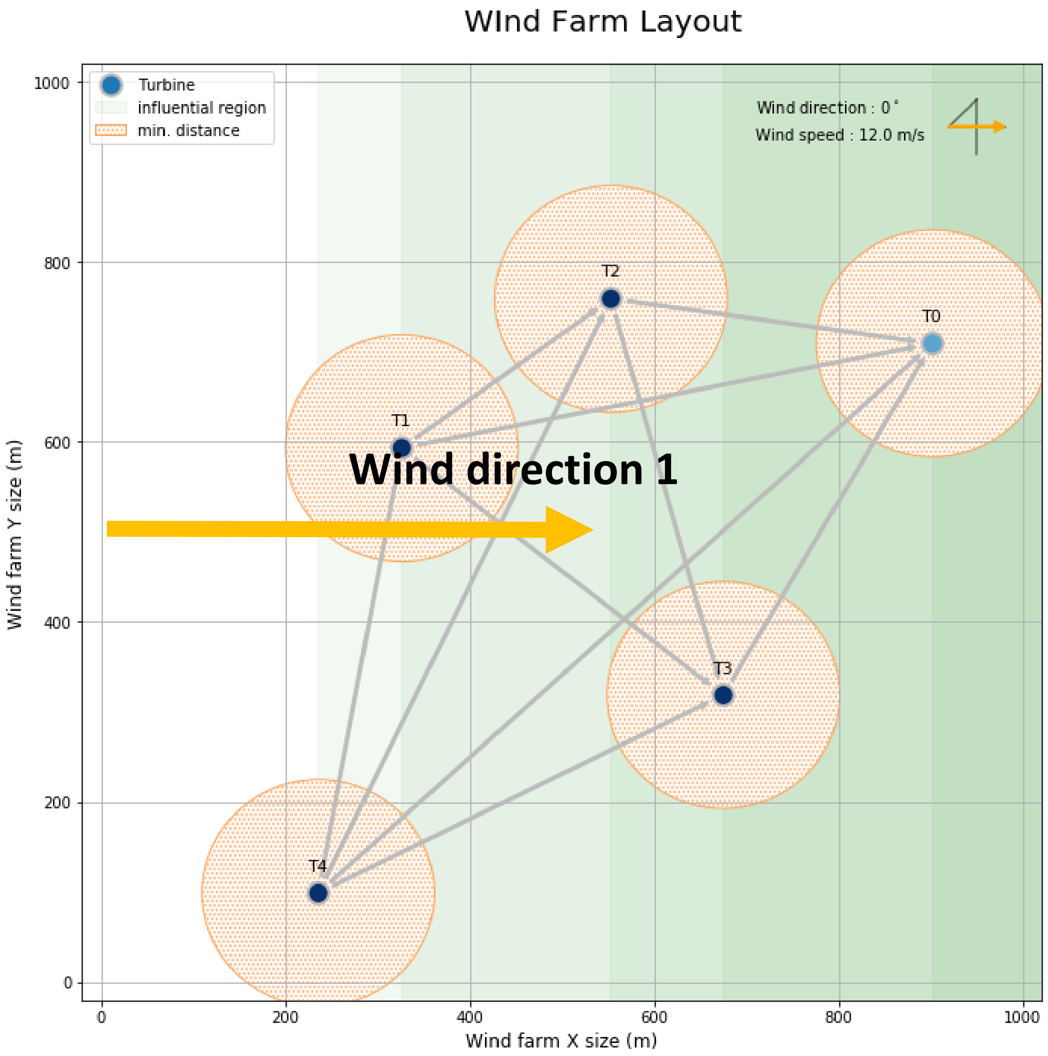
Edge features E = $\left\{\begin{pmatrix}\text{the down-stream wake distance } d, \\ \text{the radial-wake distance } r\end{pmatrix}\right\}_{\forall (i,j)^*}$

Global features g = $\{free\ flow\ wind\ speed\}$

$i, j\ are\ turbine\ index$

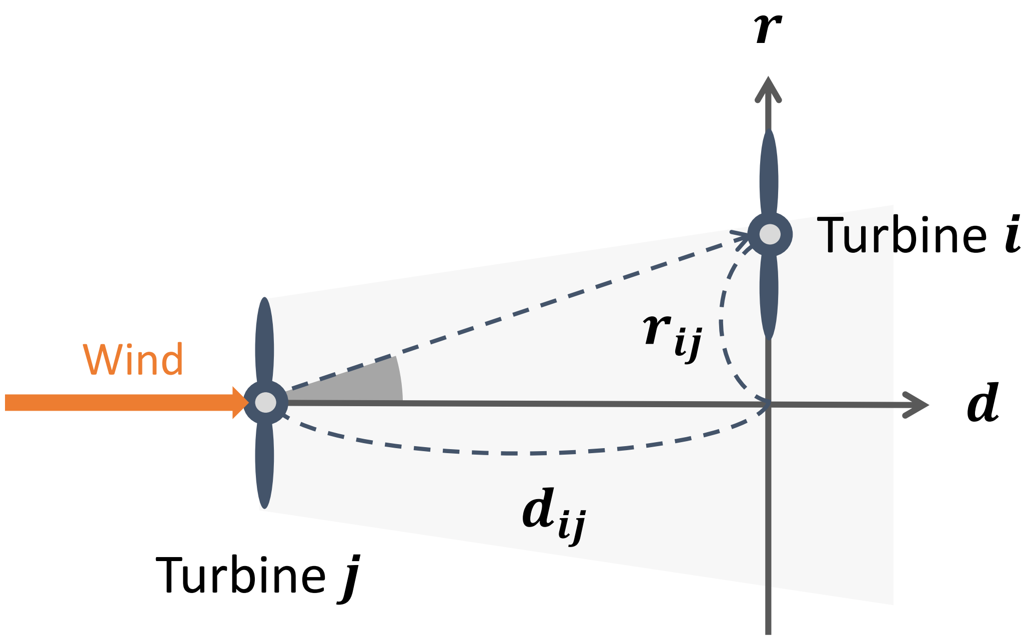
$* \ \forall (i,j)\ \in interaction\ turbines$

Details on Edge Features

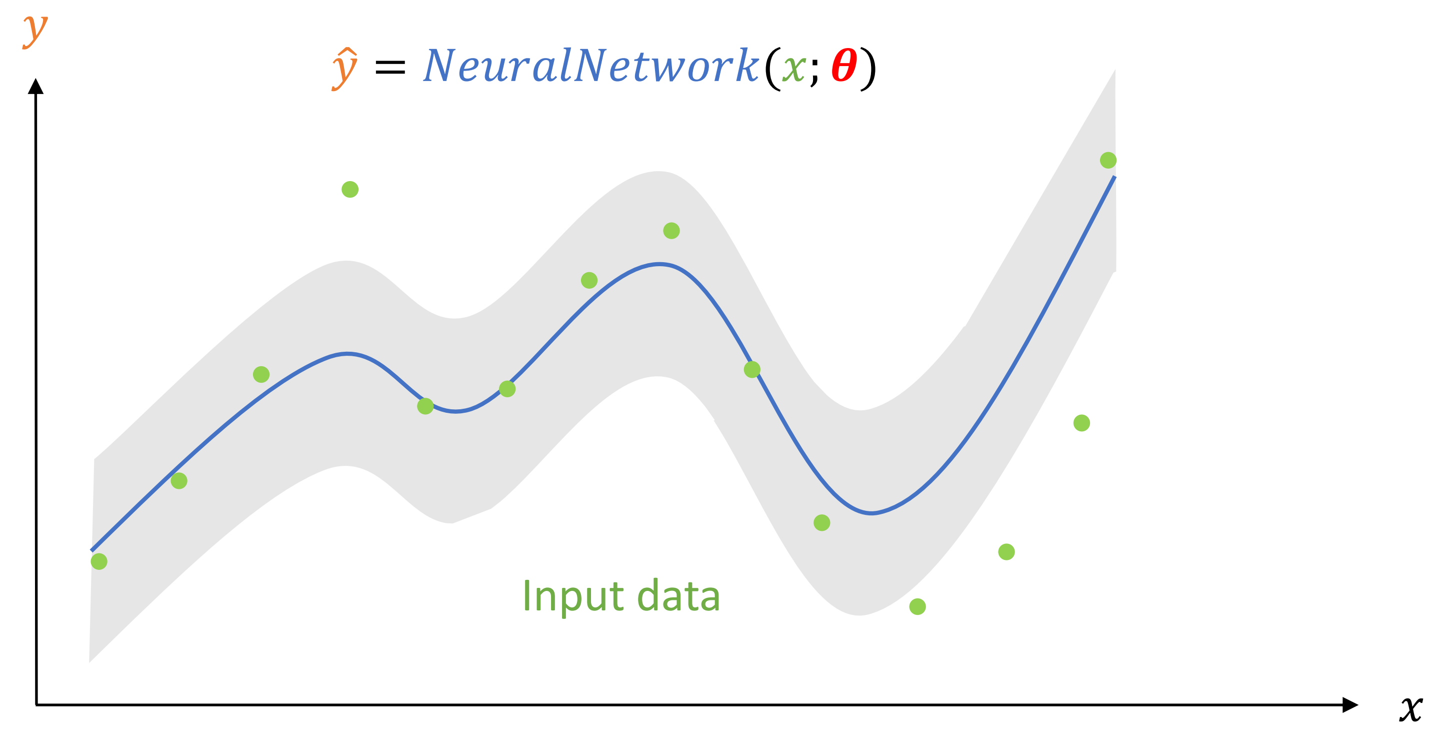


$$\mathcal{G} = (N, E, g)$$

Edge features E $= \left\{ \begin{pmatrix} \text{the down-stream wake distance } \boldsymbol{d}, \\ \text{the radial-wake distance } \boldsymbol{r} \end{pmatrix} \right\}_{\forall (i,j)^*}$



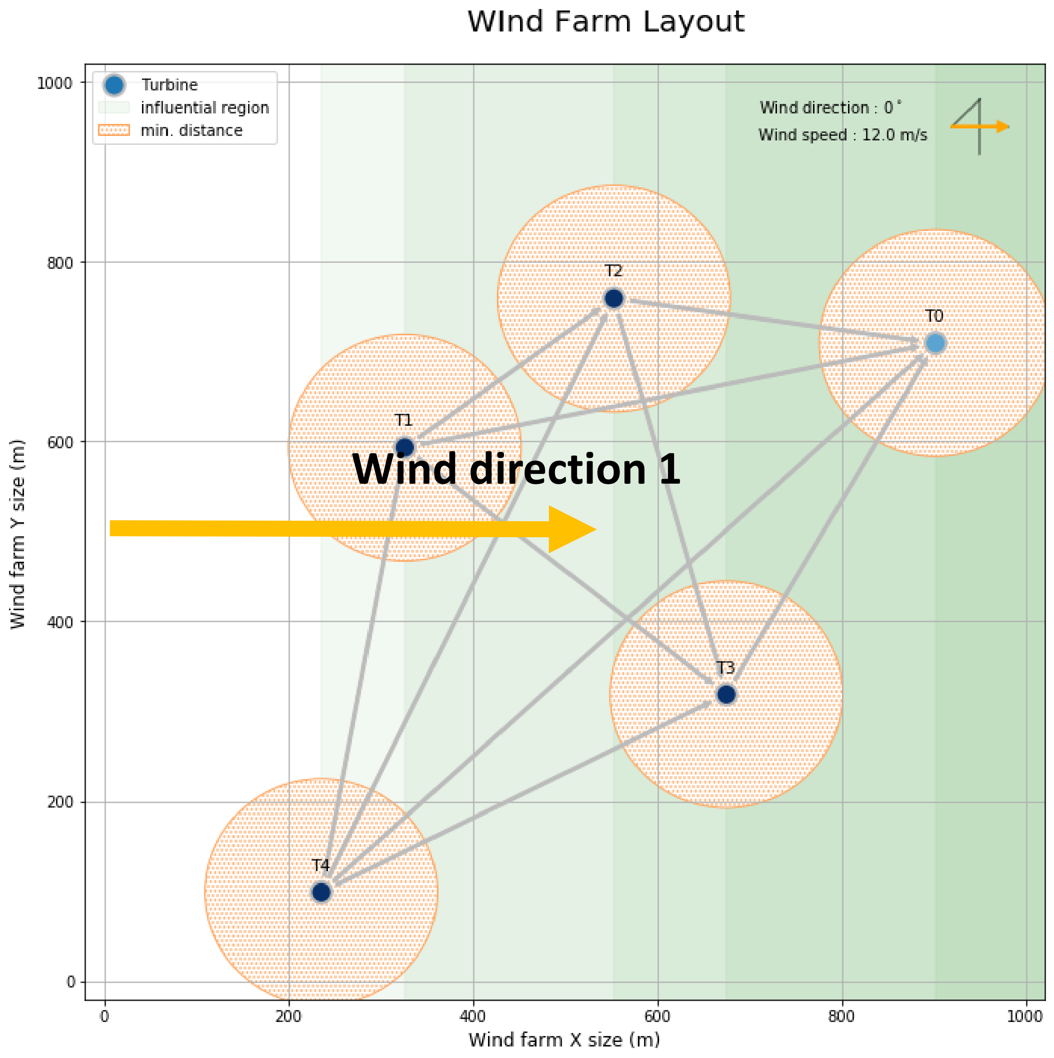
Neural Network in EXTREMELY High Level View



Neural network is a function approximator that has trainable parameter $\boldsymbol{\theta}$

such that $y \approx \hat{y}$ as accurate as possible

Why Graph Representation?



$$\mathcal{G} = (N, E, g)$$

vs.

#. Turbines

	X coord.	Y coord.
T0	850	713
T1	303	587
T2	569	775
T3	642	290
T4	217	97

Matrix (Tensor) Representations

Why Graph Representation?

#. Turbines

	X coord.	Y coord.
T0	850	713
T1	303	587
T2	569	775
T3	642	290
T4	217	97

1. MLP/CNN's input size tends to be fixed.

   e.g.) MNIST = [28 X 28]

   If we deploy one more turbine to the farm,

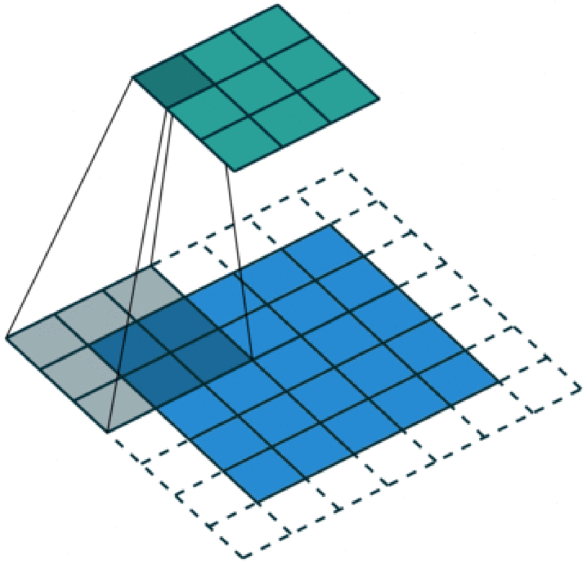
   then the input dimension would change

2. Input data has no natural order.

   e.g.) time-series has time index!

   Which turbine should be the first input?

Spatial/Temporal Adjacency does not imply 'related'



Convolution operation presumes that

'Nearby pixels are somewhat related'.

Since we ***share*** the convolution filters

RNNs presumes that

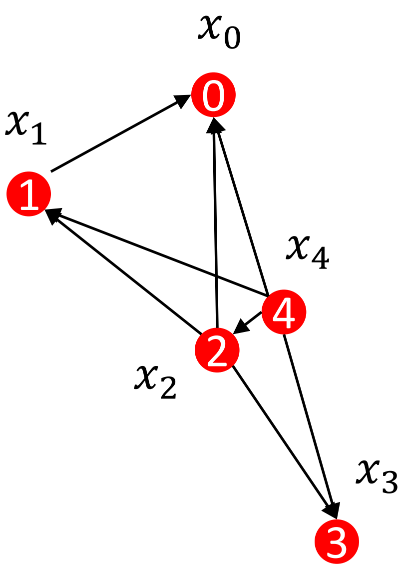
'Nearby inputs are somewhat related'.

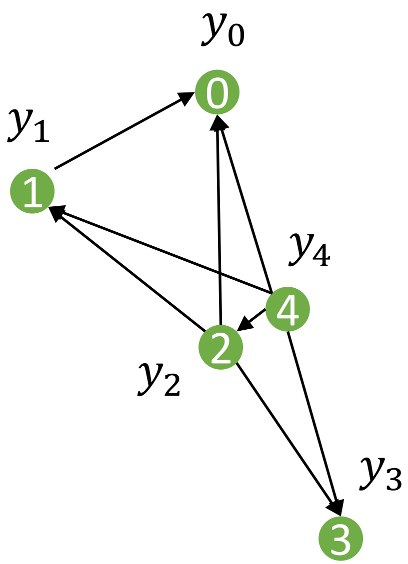
Since we ***share*** the RNN blocks.

Figure source <Left: https://github.com/vdumoulin/conv_arithmetic>, <Right: https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9>

Graph Neural Network

$$\mathcal{G}_y = GraphNeuralNetwork\,(\mathcal{G}_x; \theta)$$



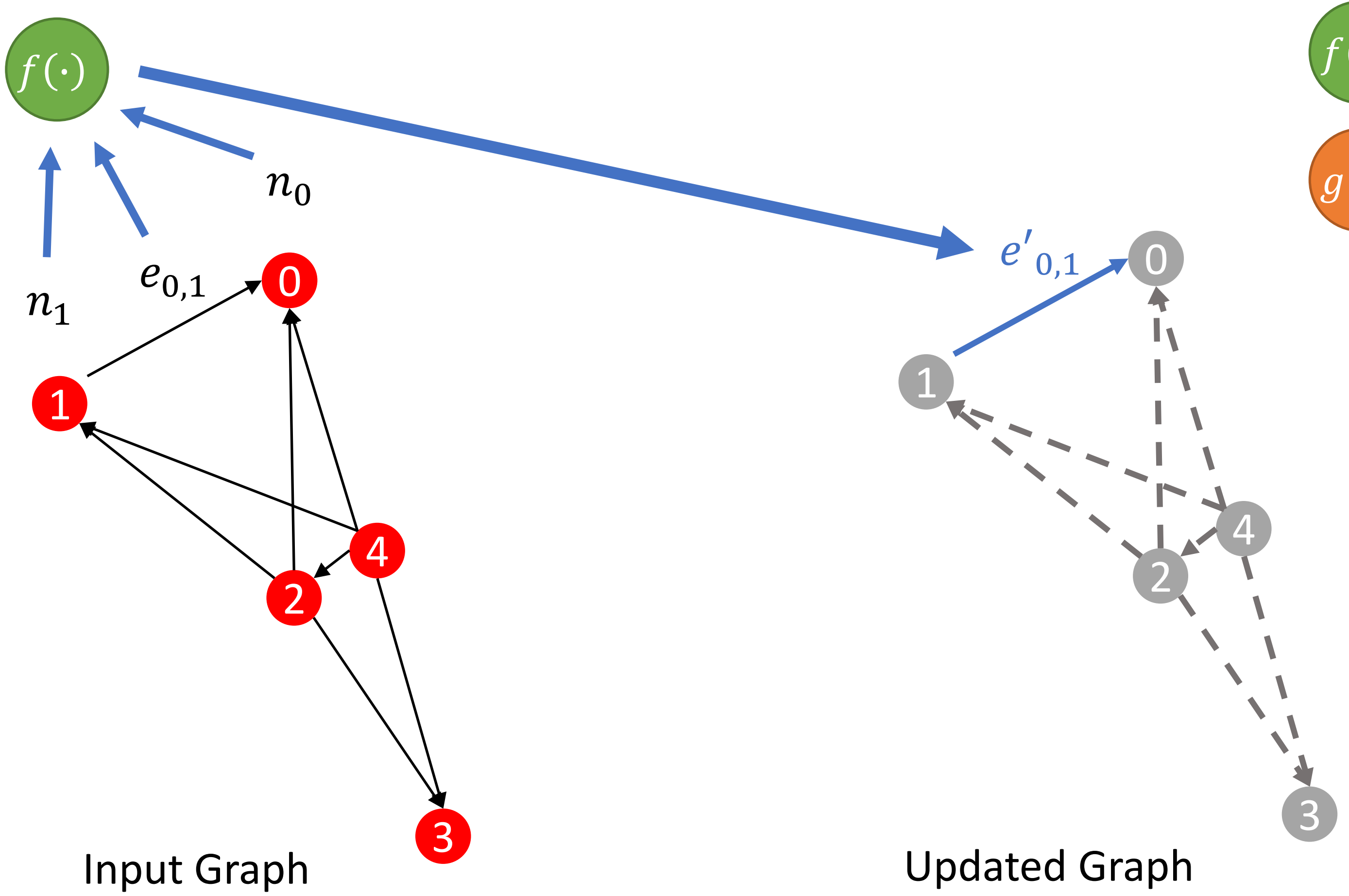


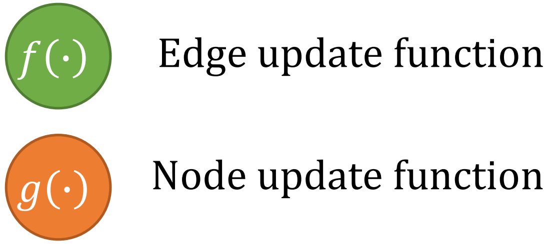
- Graph Convolution Networks (GCN)
- Attention based approaches
- Relational inductive bias (GN block)
- …

(or tensors)

Image source <https://becominghuman.ai/lets-build-a-simple-neural-net-f4474256647f?gi=743618029571>

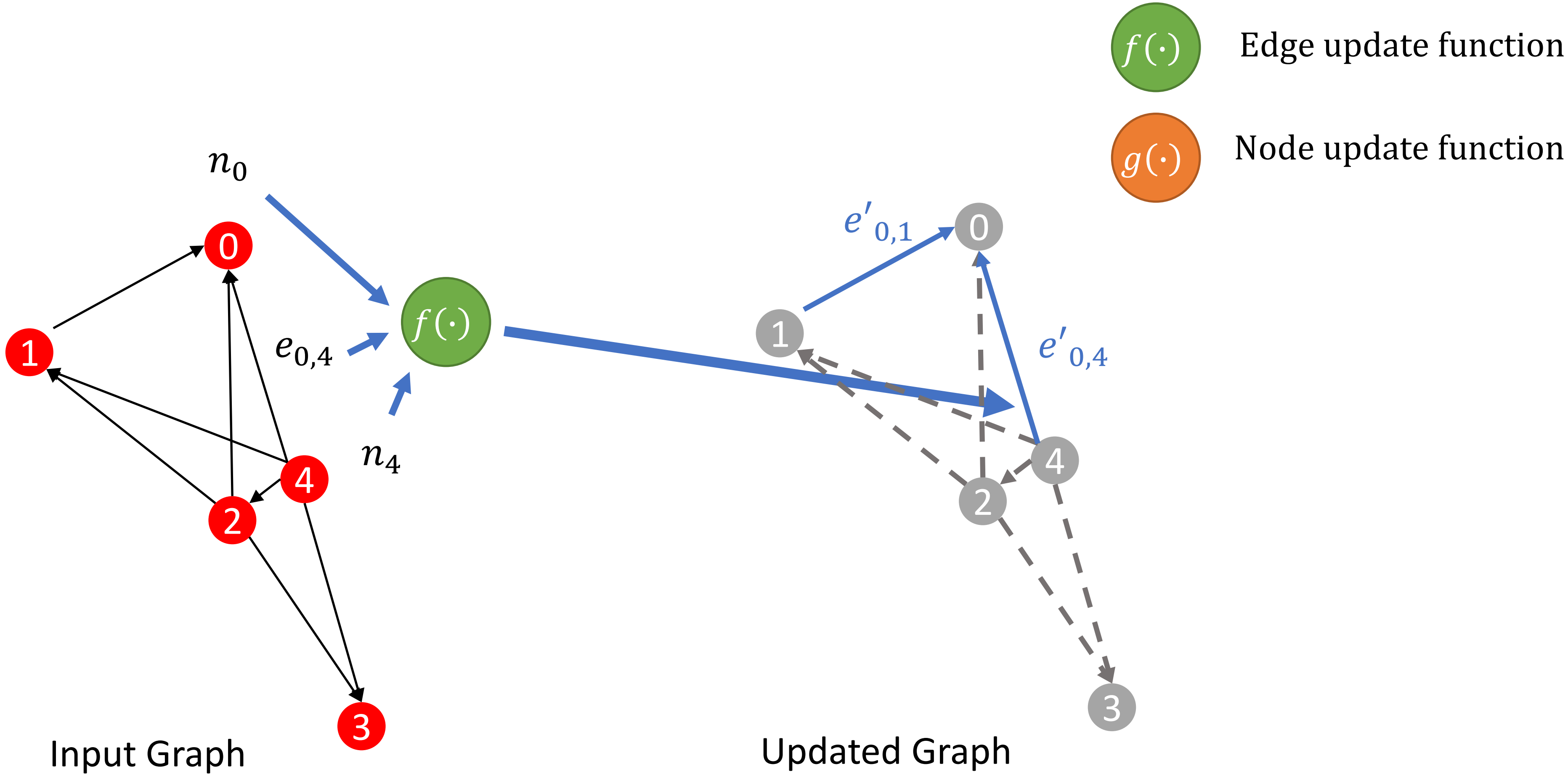
Imposing Relational Inductive Bias





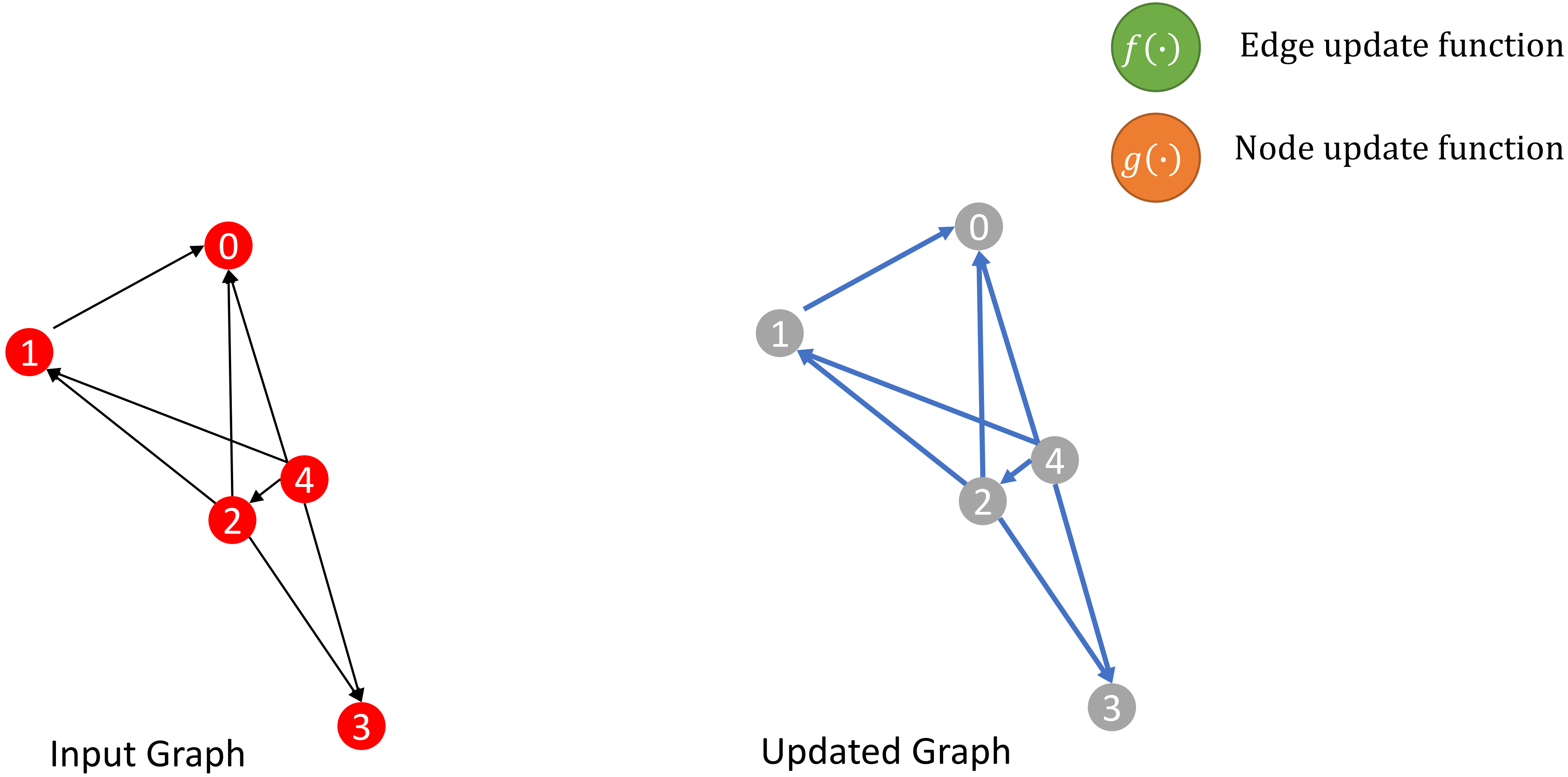
<u>Share</u> edge update function f and node update function g
for updating graph represented data

Imposing Relational Inductive Bias



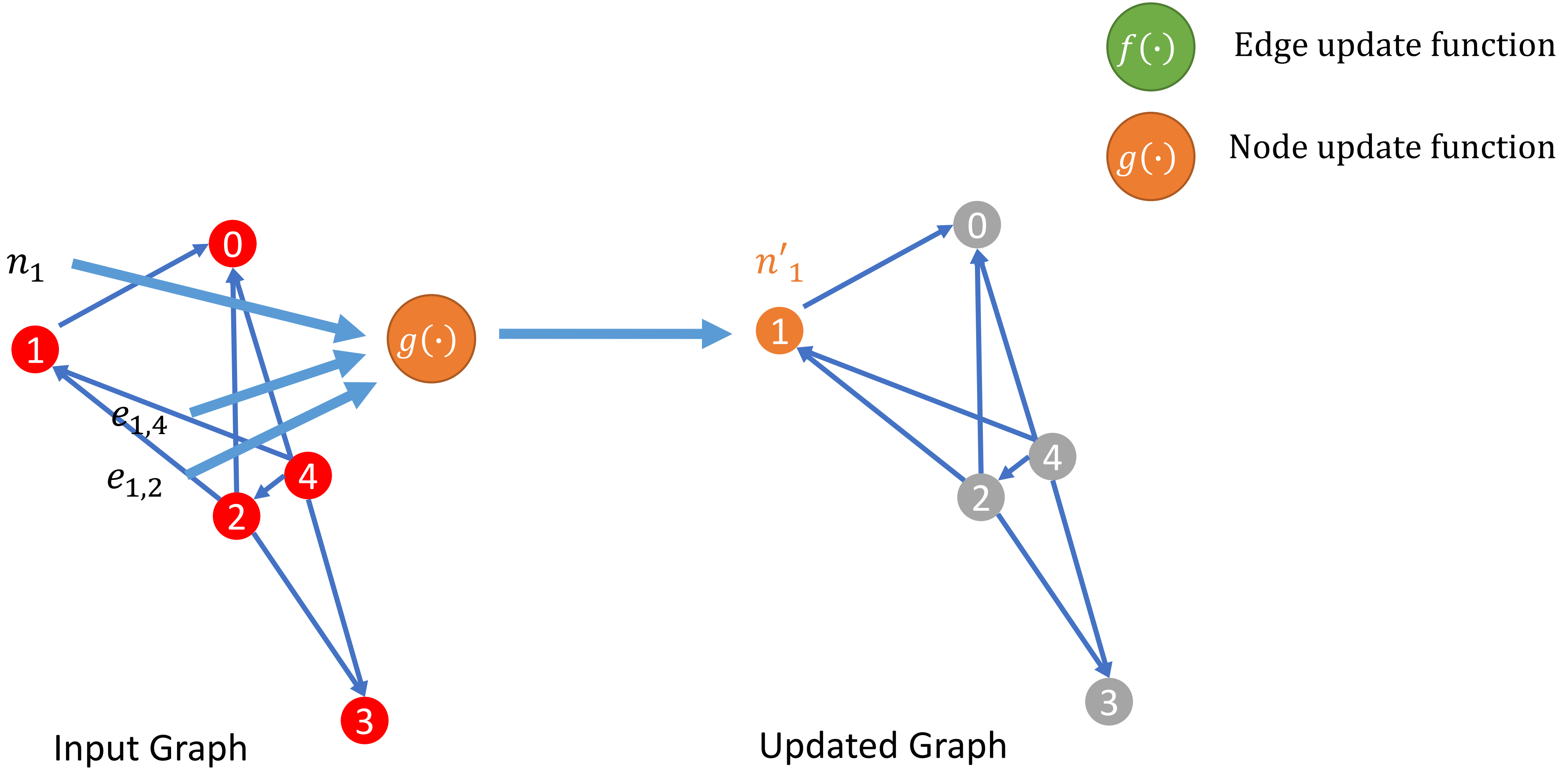
Share edge update function f and node update function g
for updating graph represented data

Imposing Relational Inductive Bias



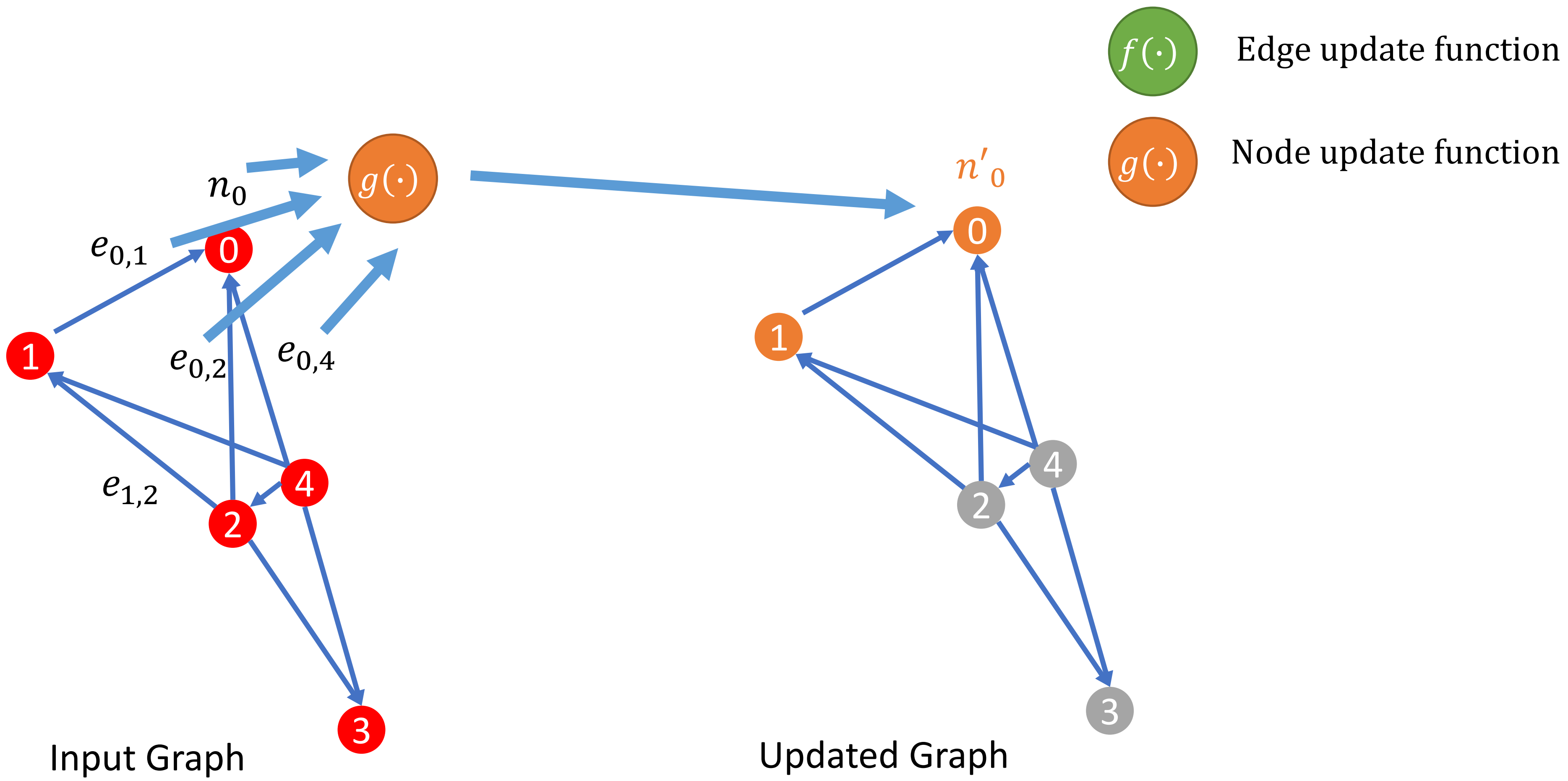
<u>Share</u> edge update function f and node update function g
for updating graph represented data

Imposing Relational Inductive Bias



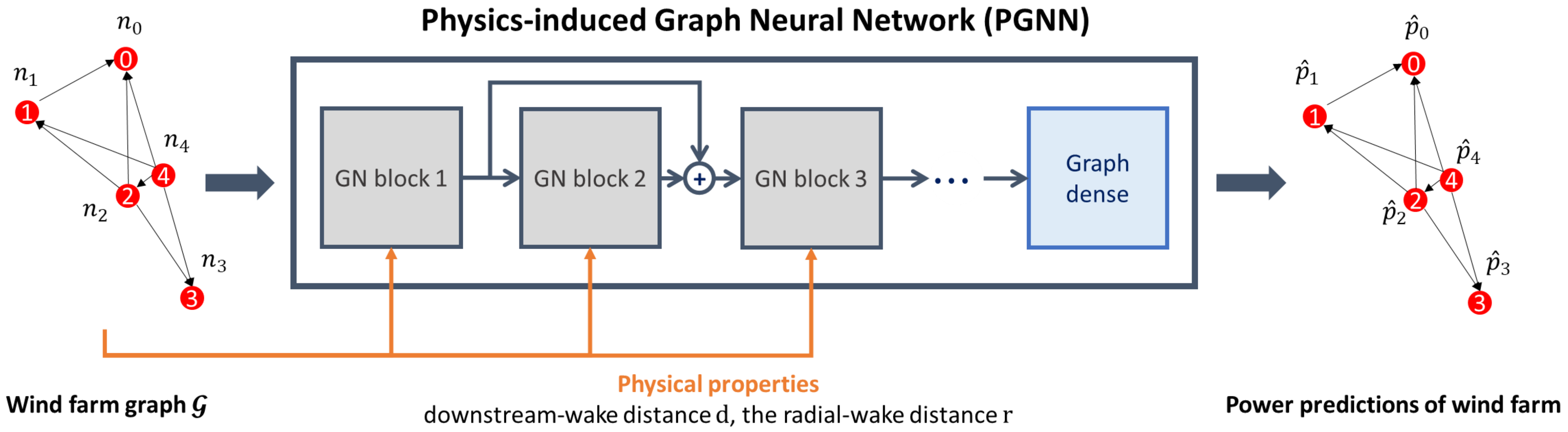
Share edge update function f and node update function g
for updating graph represented data

Imposing Relational Inductive Bias

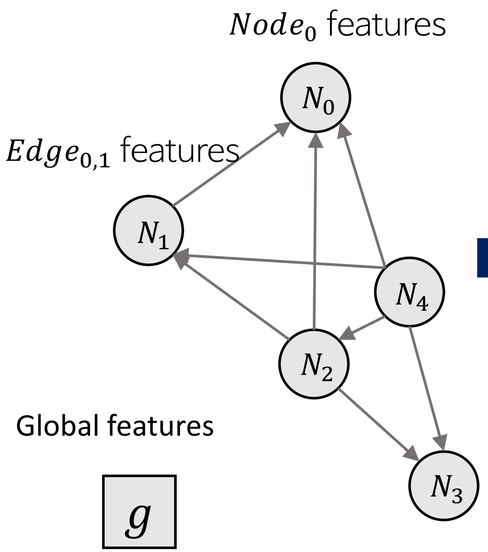


Share edge update function f and node update function g
for updating graph represented data

Physics-induced Graph Neural Network On Wind Power Estimations



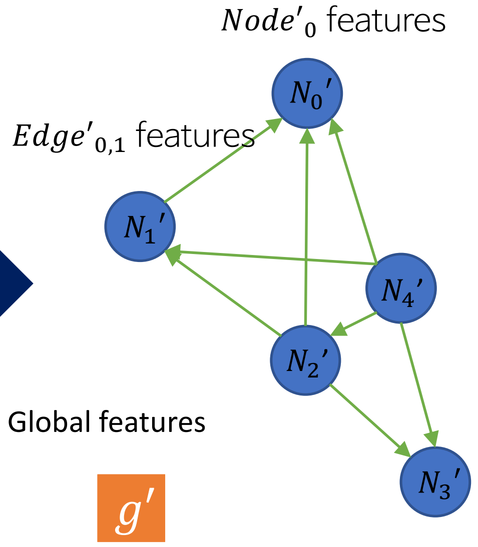
GN (Graph Neural) Block



Input graph $\mathcal{G}$

Graph Neural (GN) Block

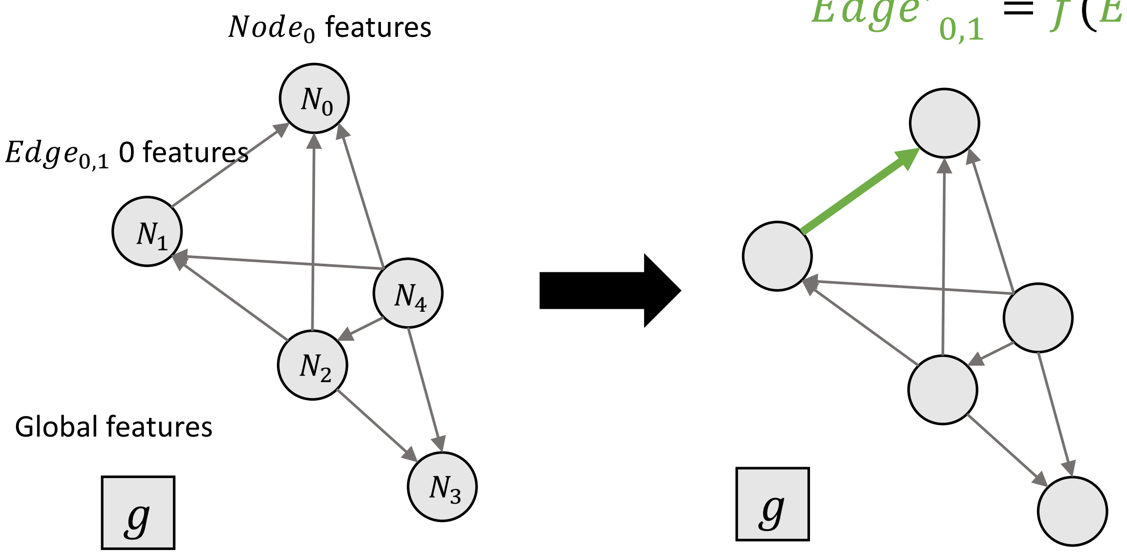
Edge update network	$f(\cdot : \theta_0)$
Node update network	$f(\cdot : \theta_1)$
Global update network	$f(\cdot : \theta_2)$



Update graph $\mathcal{G}'$

GN Block – Edge update steps

$Edge'_{0,1} = f(Edge_{0,1}\ Node_1, Node_0, g; \theta_0)$

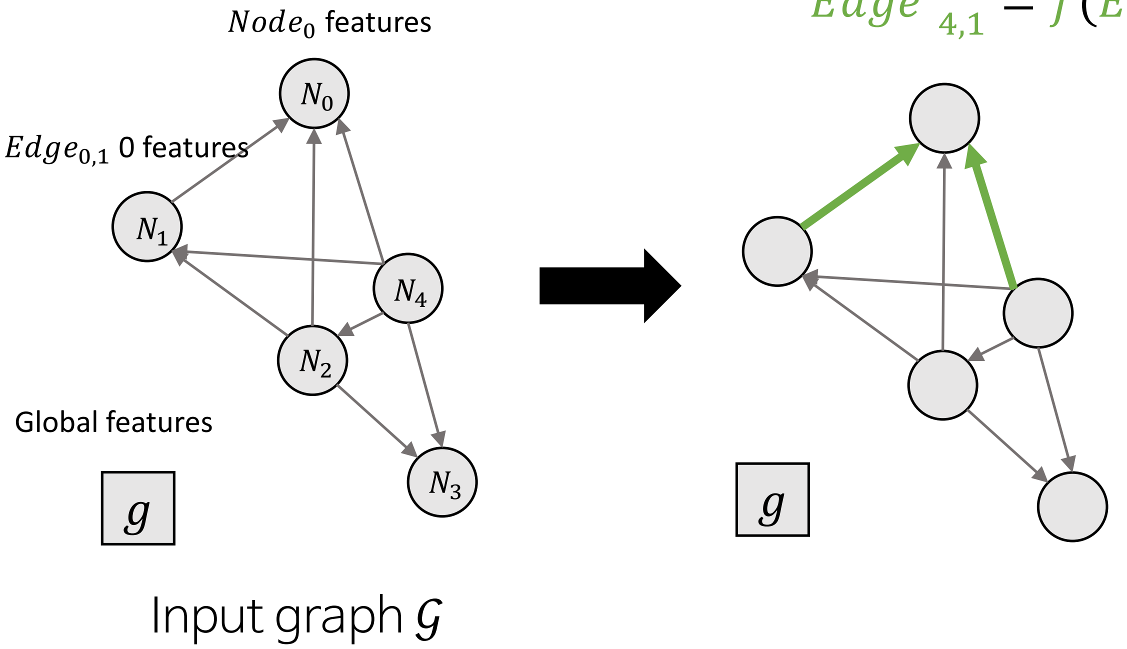


Input graph $\mathcal{G}$

Update edge features with $f(Edge\ features, Reciever\ features, Sender\ features, g; \theta_0)$

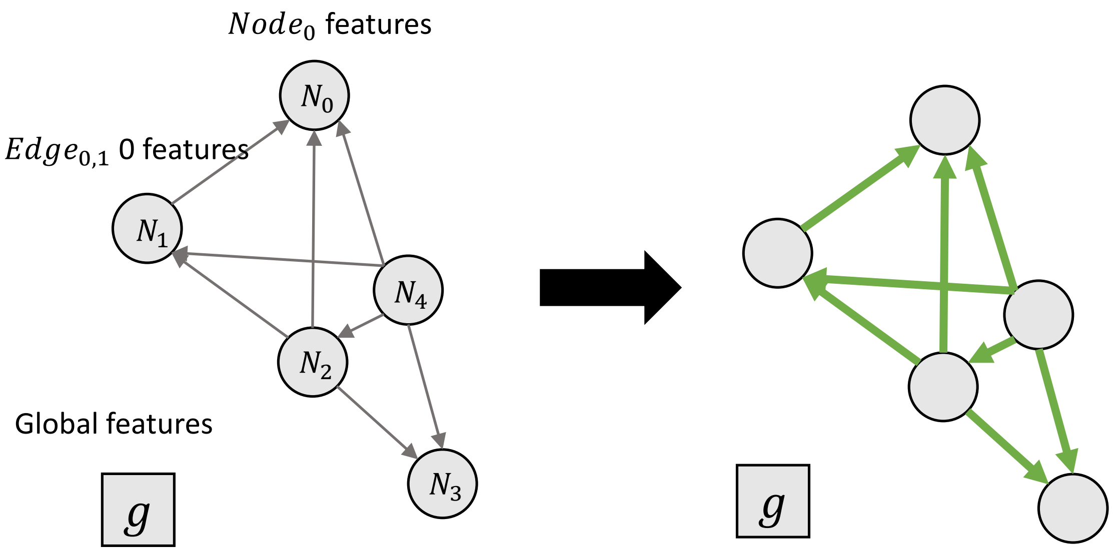
GN Block – Edge update steps

$Edge'_{4,1} = f(Edge_{4,1}\ Node_4, Node_1, g; \theta_0)$



Update edge features with $f(Edge\ features, Reciever\ features, Sender\ features, g; \theta_0)$

GN Block – Edge update steps



Input graph $\mathcal{G}$

Updated edge features

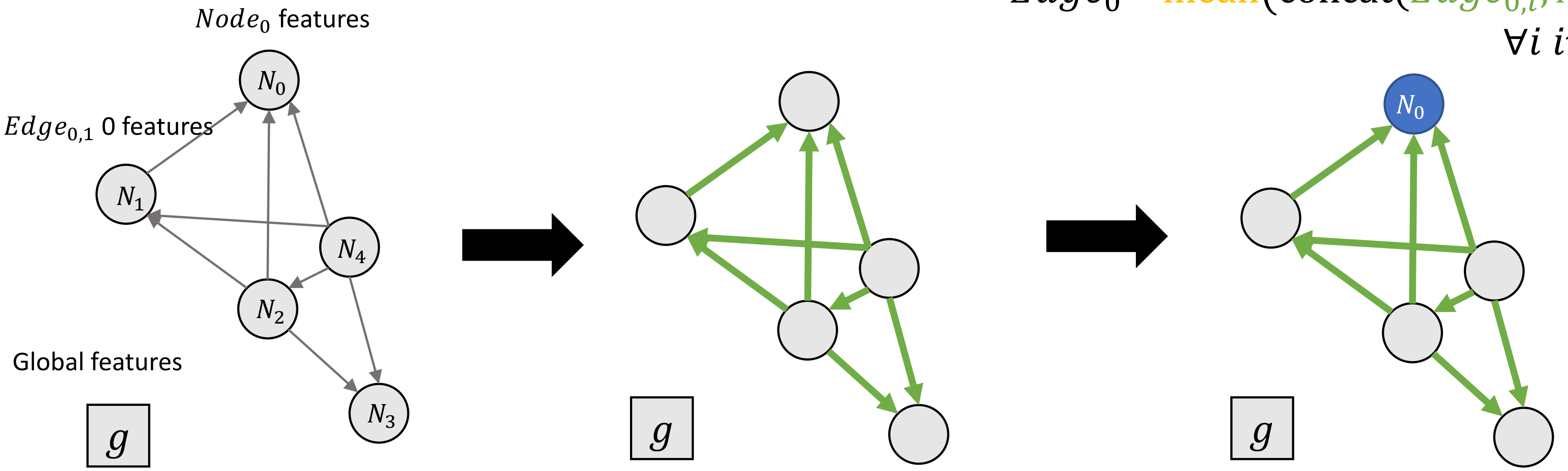
Update edge features with $f(Edge\ features, Reciever\ features, Sender\ features, g; \theta_0)$

GN Block – Node update steps

$Node'_0 = f(Edge_0; \theta_1)$

$Edge_0$ = $\text{mean}\big(\text{concat}(Edge_{0,i}, Node_1, Node_i)\big)$

$\forall i \text{ incoming edges}$

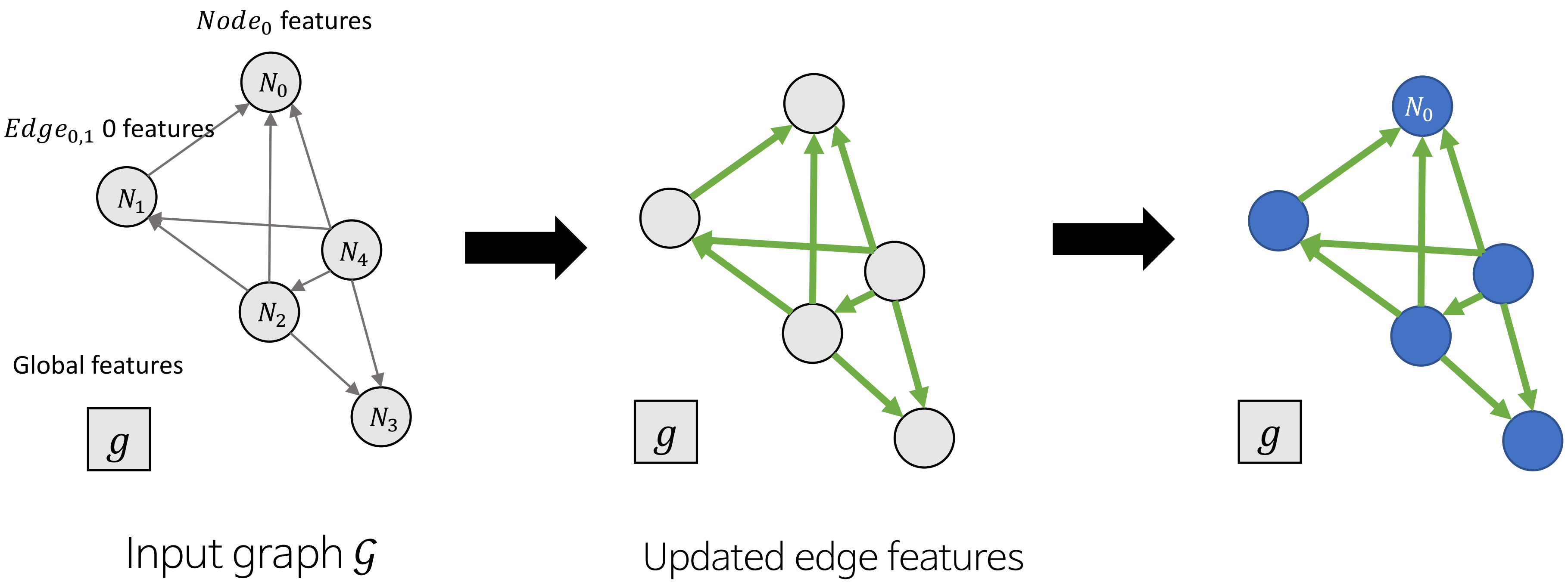


Input graph $\mathcal{G}$

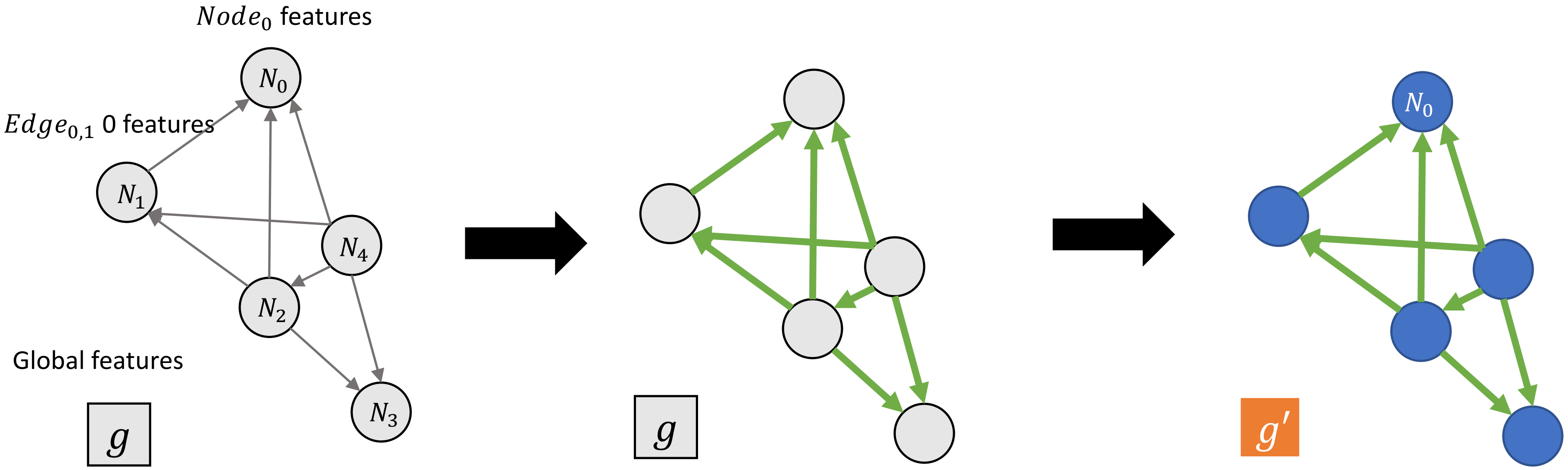
Updated edge features

Aggregation function: any function obeys 'input-order invariant' and 'input-number invariant' properties.
e.g., Mean, Max, Min, etc.

GN Block – Node update steps



GN Block – Global feature update



Input graph $\mathcal{G}$

Updated edge features

$$g' = f(Edge', Node', g; \theta_2)$$

$$Edge' = \text{mean}(Edge'_{i,j})\ \forall edges\ i, j$$

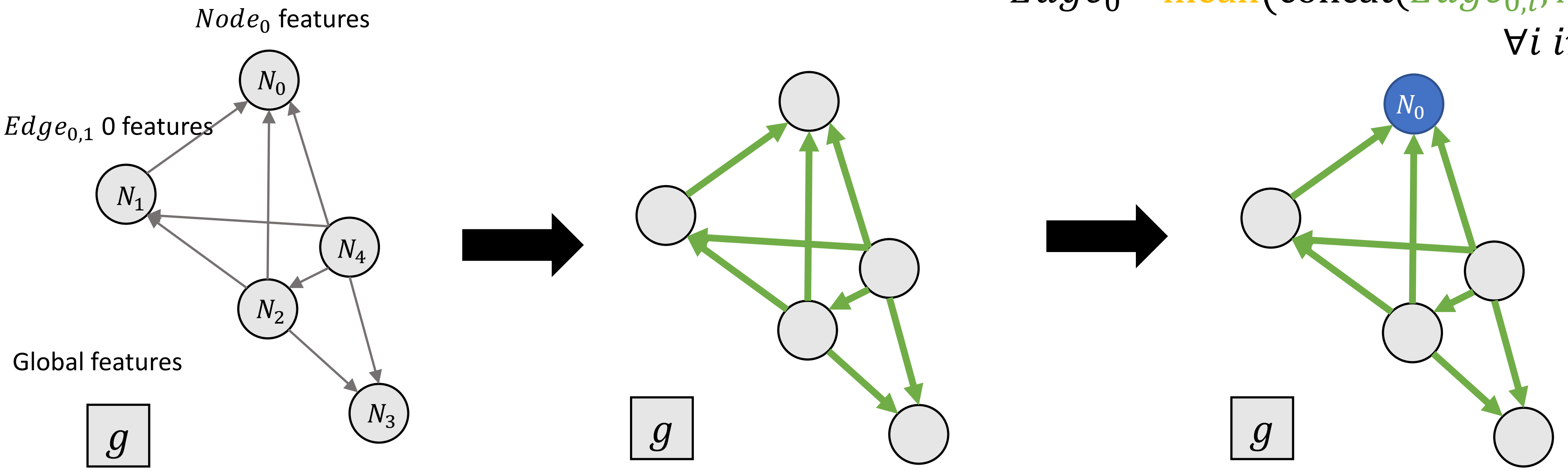
$$Node' = \text{mean}(Node_i)\ \forall\ nodes\ i$$

Revisit Aggregation Method

$$Node'_0 = f(Edge_0; \theta_1)$$

$$Edge_0 = \text{mean}\big(\text{concat}(Edge_{0,i}, Node_1, Node_i)\big)$$

$\forall i \; incoming \; edges$



Input graph $\mathcal{G}$

Updated edge features

Aggregation function: any function that obeys 'input-order invariant' and 'input-number invariant' properties.
e.g., Mean, Max, Min, etc.

Weighted "__" ≈ Attention (in Deep Learning)

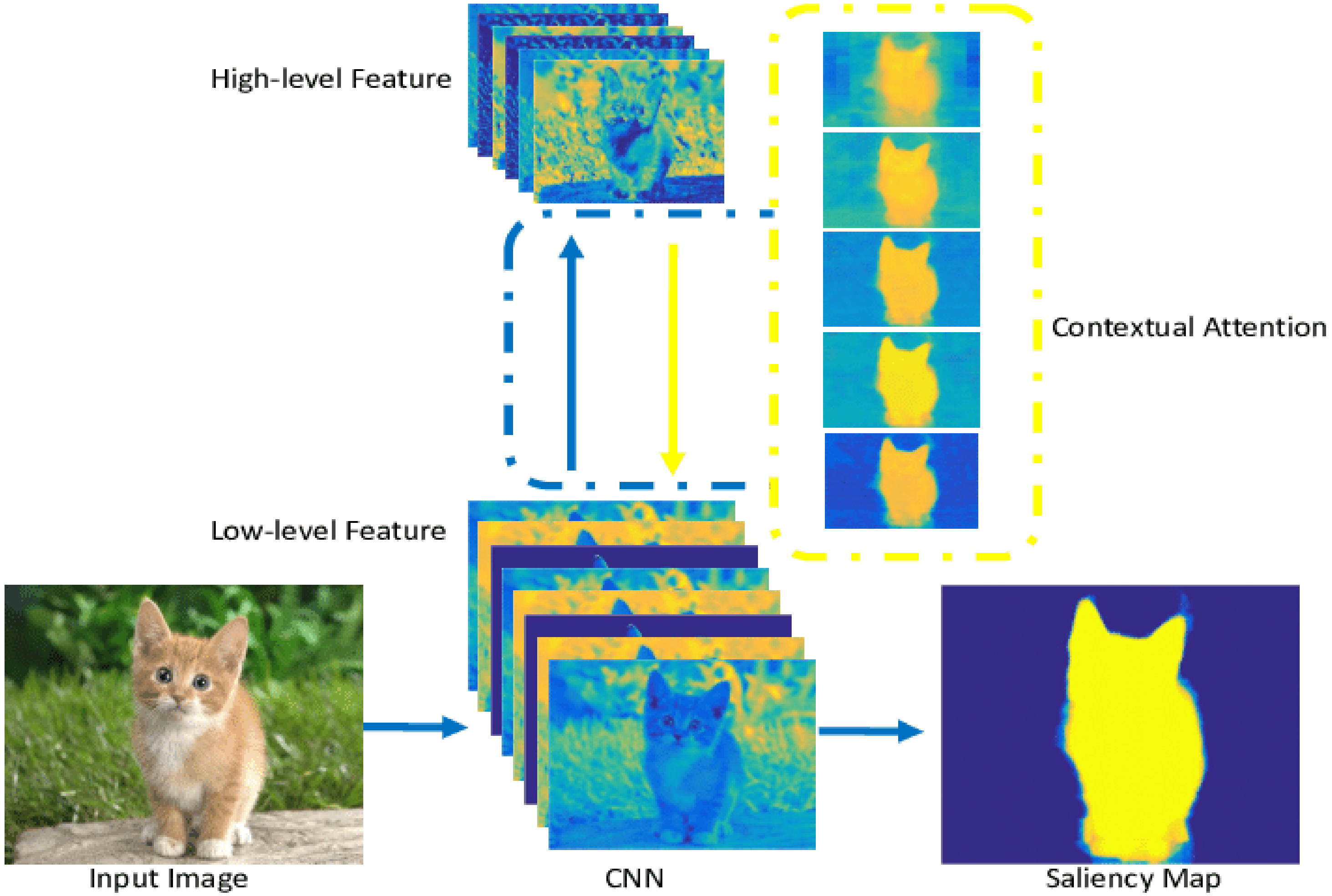
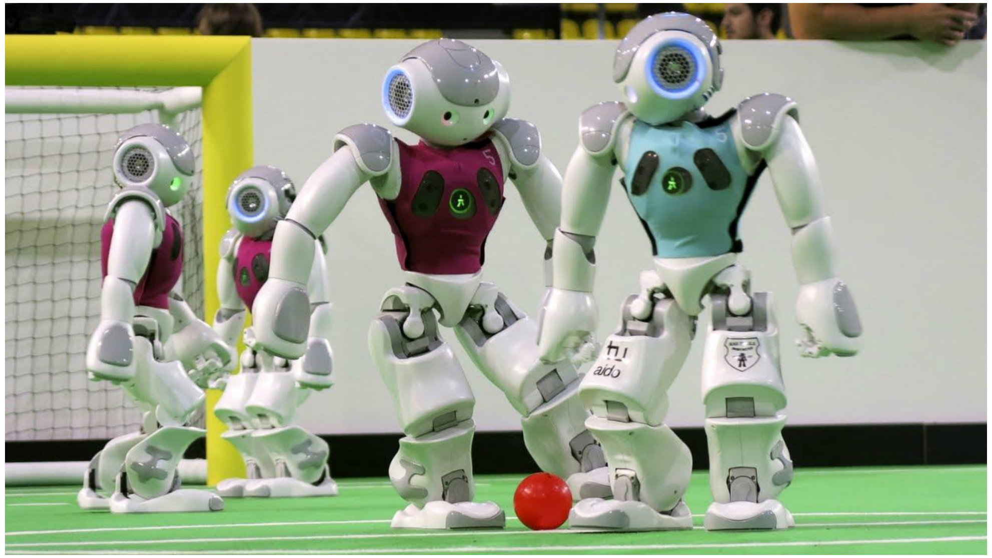
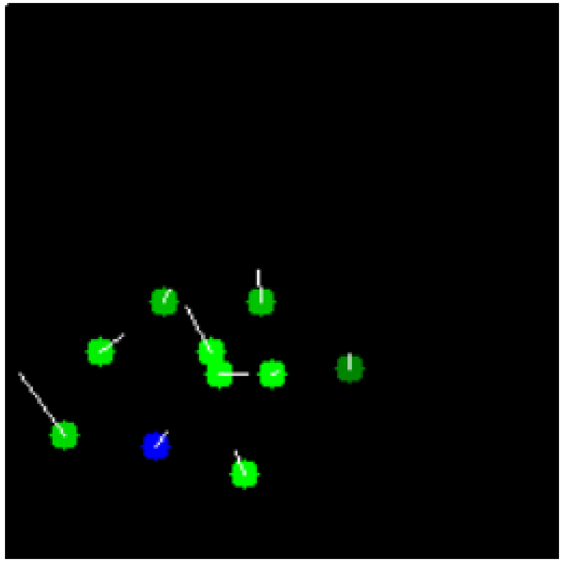


Figure source <Agile Amulet: Real-Time Salient Object Detection with Contextual Attention>

Consider weighted Aggregations



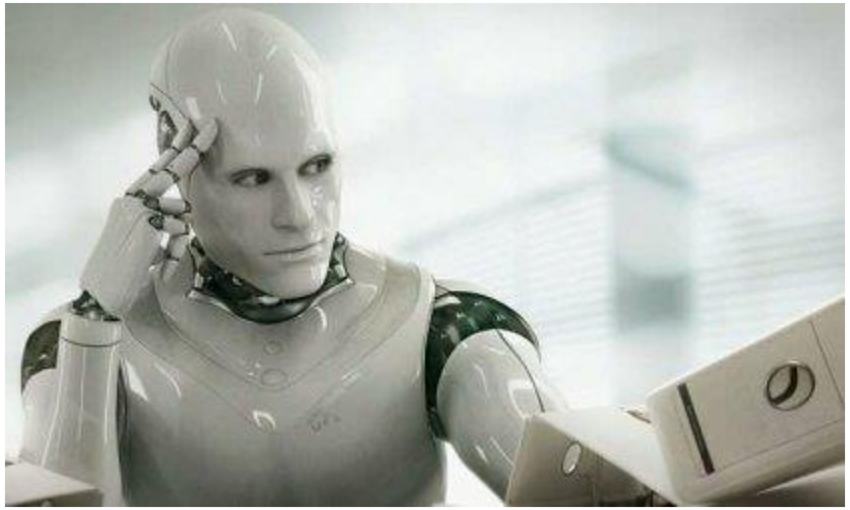
<Robot soccer>



<Visualized weights>

Figure source <Left: https://www.youtube.com/watch?v=HHlN0TDglIE> , <Right: VAIN: Attentional Multi-agent Predictive Modeling>

How can we get the weights?

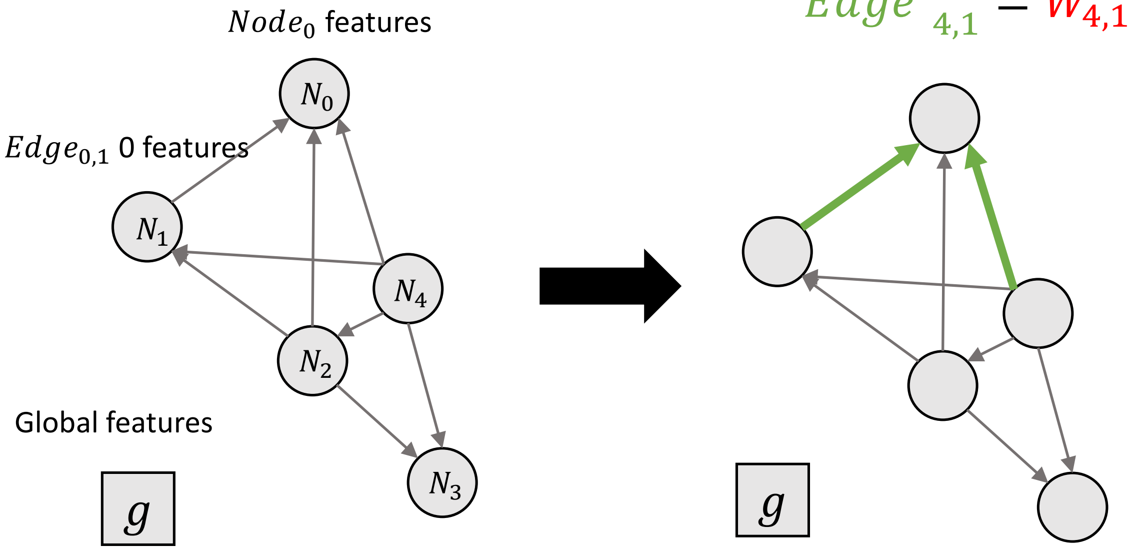


Learn to weight!

GN Block – Edge update steps Revisit

$$W_{4,1} = f(some\ possible\ inputs; \theta_3)$$

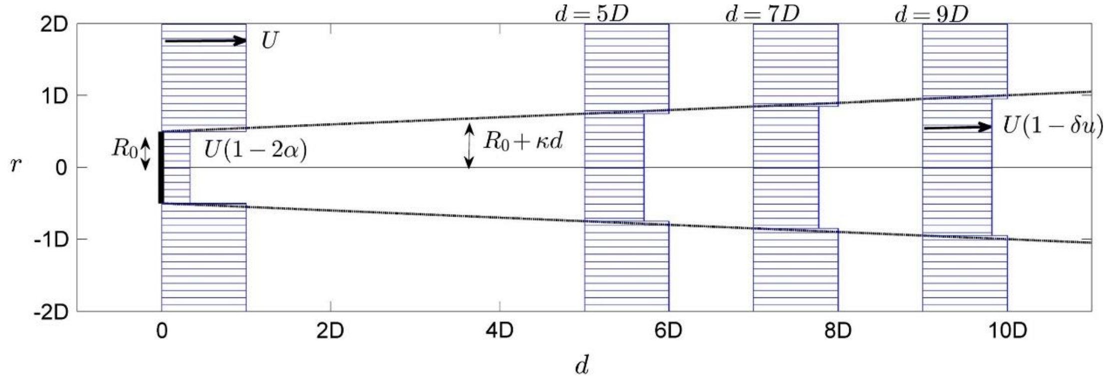
$$Edge'_{4,1} = W_{4,1} \times f(Edge_{4,1}\ Node_4, Node_1, g; \theta_0)$$



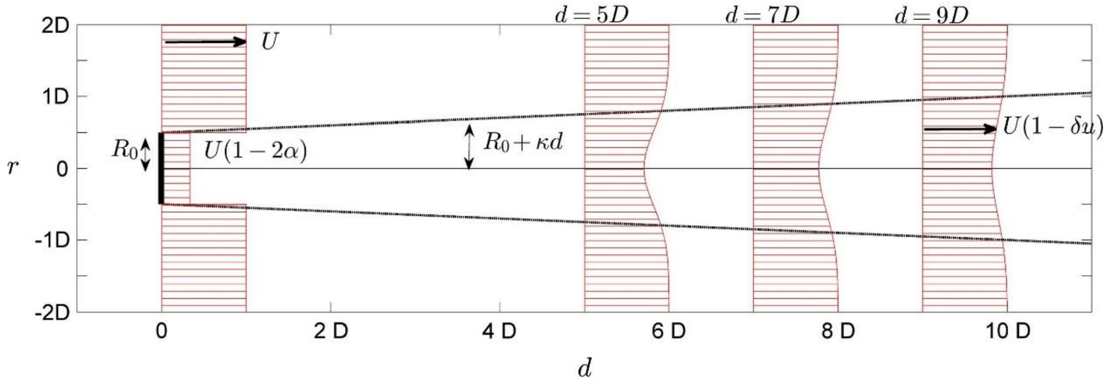
Input graph $\mathcal{G}$

Update edge features with $f(Edge\ features, Reciever\ features, Sender\ features, g; \theta_0)$

Physics-induced Attention



(a) Park wake model (Jensen, 1983)



(b) Continuous wake model

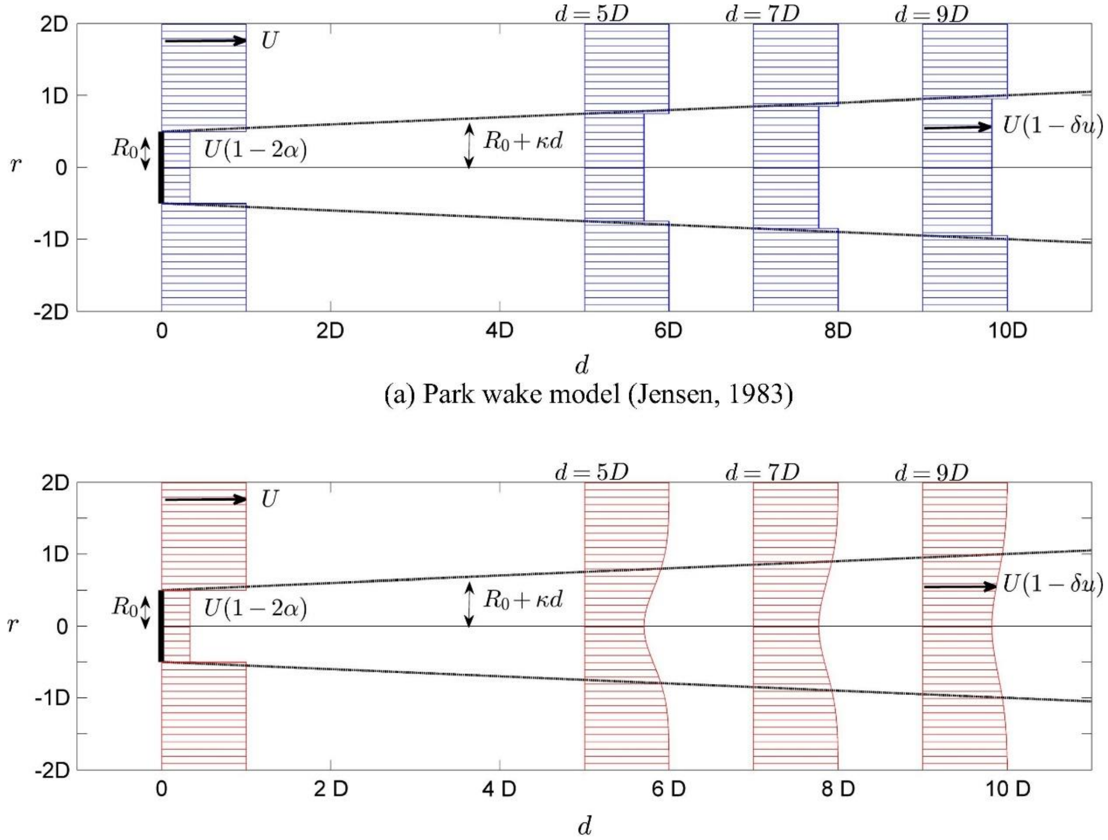
JK park, and K.H. law suggest the continuous deficit factor δu(d, r, α) as

$$\delta u(d, r) = 2\alpha\left(\frac{R_0}{R_0+\kappa d}\right)^2 \exp\left(-\left(\frac{r}{R_0+\kappa d}\right)^2\right)$$

R_0: *Rotor diameter*
d: *Down−stream wake distance*
r: *Radial wake − distnance*
α, κ: *Tunable parameters*

Figure source <Cooperative wind turbine control for maximizing wind farm power using sequential convex programming by **Jinkyoo Park**, Kincho H.Law >

Physics-induced Attention



(a) Park wake model (Jensen, 1983)

(b) Continuous wake model

$$\delta u(d, r) = 2\alpha \left(\frac{R_0}{R_0+\kappa d}\right)^2 \exp\left(-\left(\frac{r}{R_0+\kappa d}\right)^2\right)$$

$\delta u(d, r)$ indicates
'How much the down stream turbine is affected
Due to the upstream turbines'
→ Weighting Factor W!

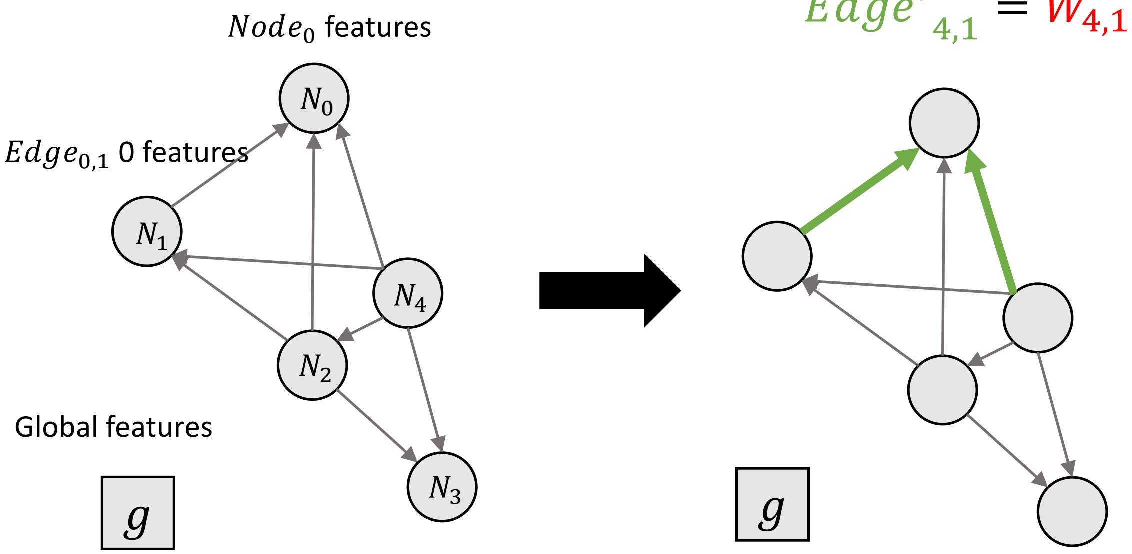
However, they tuned the parameters α, κ to the observed data

Figure source <Cooperative wind turbine control for maximizing wind farm power using sequential convex programming by **Jinkyoo Park**, Kincho H.Law >

Physics-induced Attention

$$W_{4,1} = f(\mathit{some\ possible\ inputs}; \theta_3)$$

$$Edge'_{4,1} = W_{4,1} \times f(Edge_{4,1}\ Node_4, Node_1, g; \theta_0)$$



Input graph $\mathcal{G}$

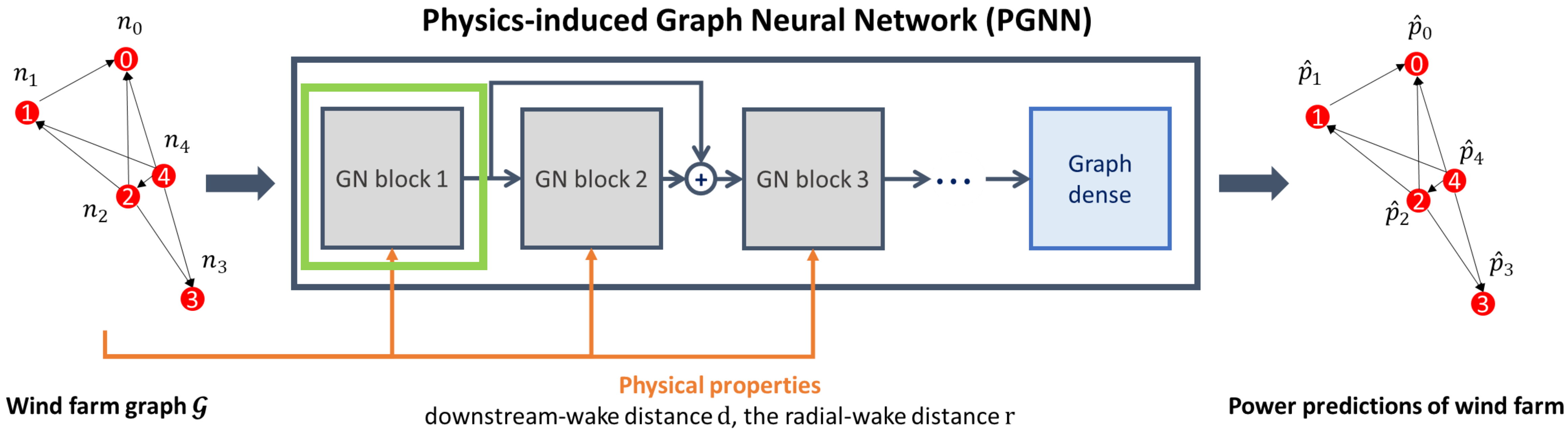
$$f(\mathit{some\ possible\ inputs}; \theta_3)$$

$$f(r, d; \alpha, \kappa, R_0)$$

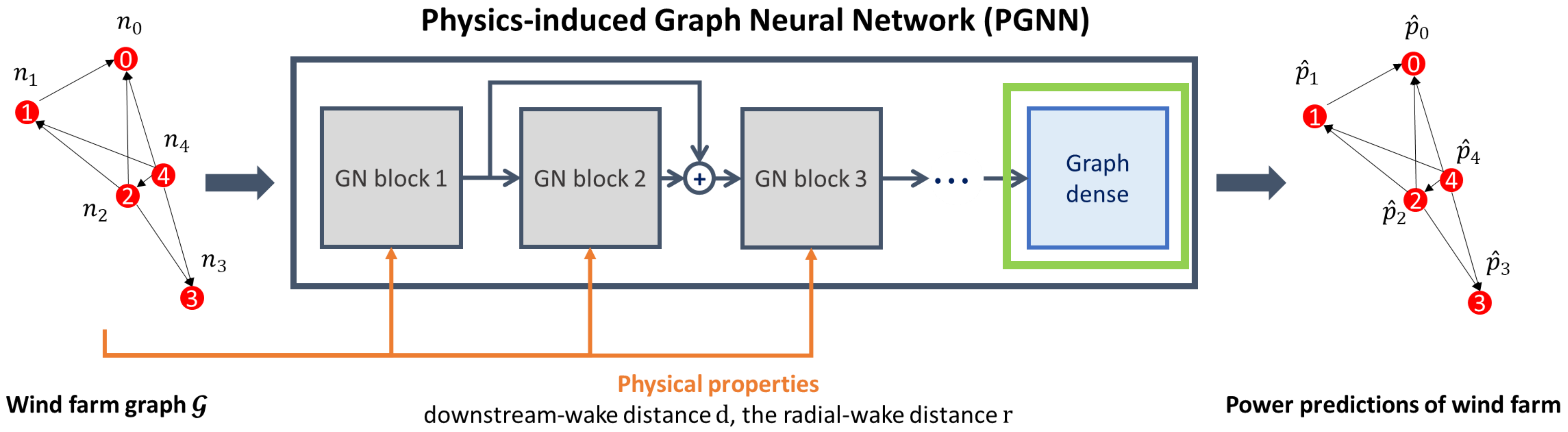
$$= 2\alpha\left(\frac{R_0}{R_0 + \kappa d}\right)^2 \exp\left(-\left(\frac{r}{R_0 + \kappa d}\right)^2\right)$$

Let neural network **learn** α, κ, R_0!

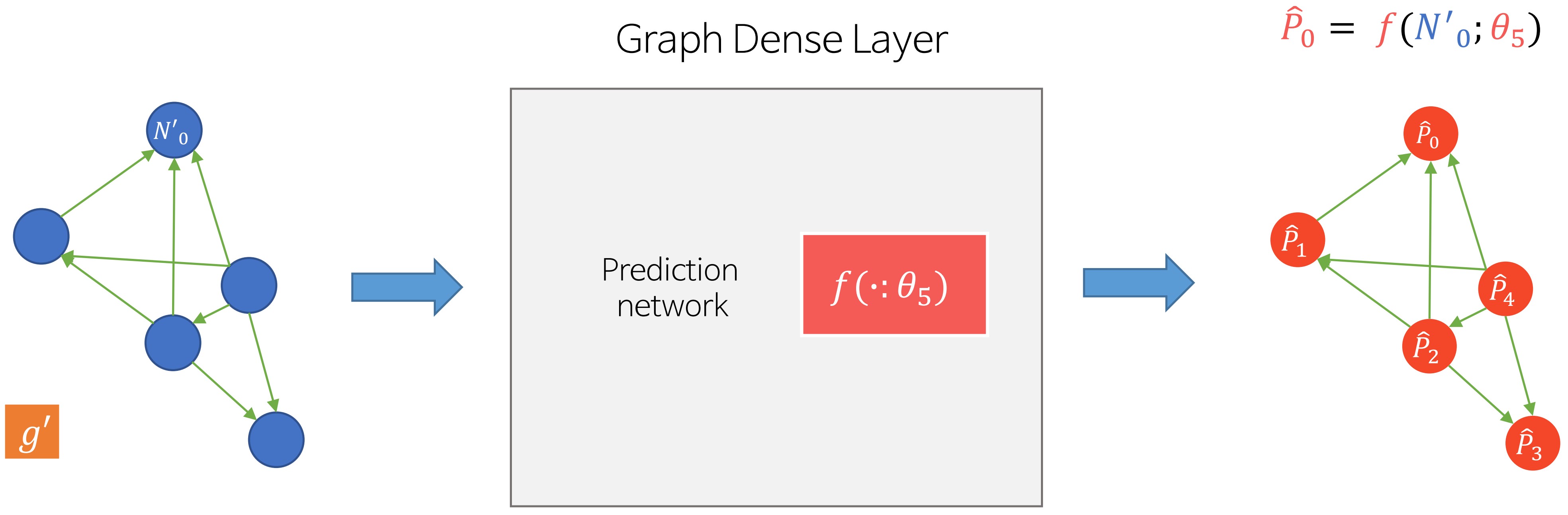
Physics-induced Graph Neural Network On Wind Power Estimations



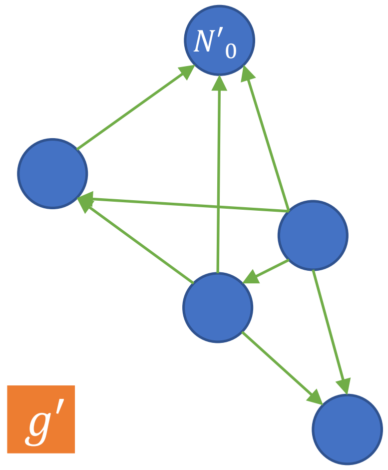
Physics-induced Graph Neural Network On Wind Power Estimations

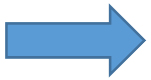


Graph Dense Layer

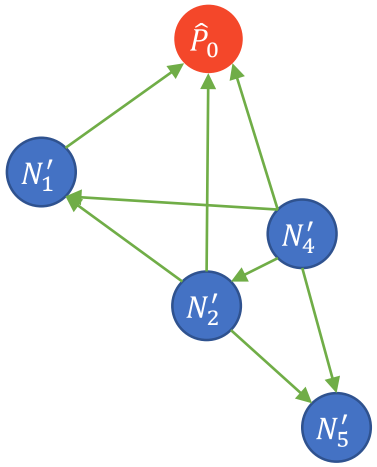


Graph Dense Layer



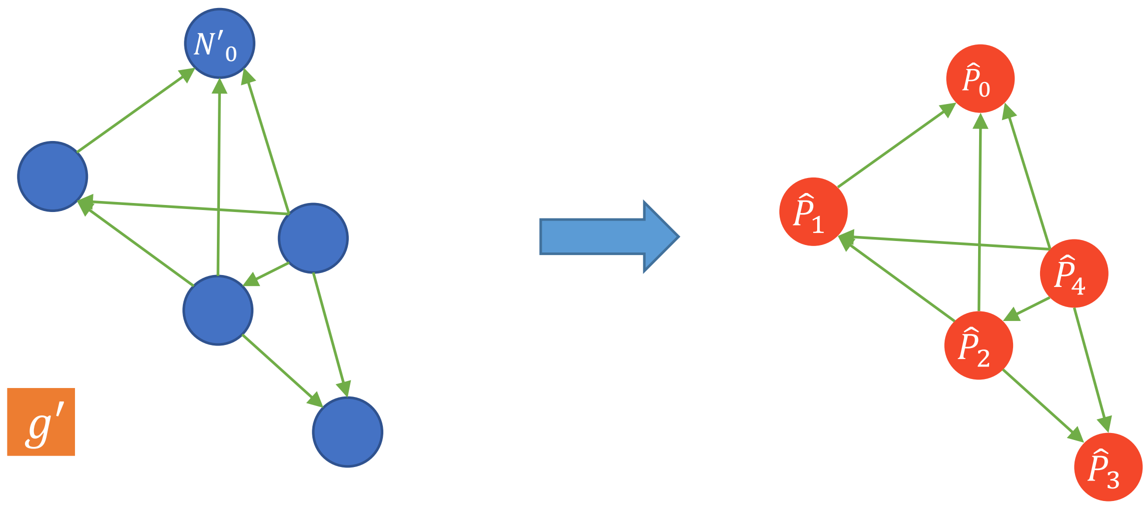


$$\hat{P}_0 = f(N'_0; \theta_5)$$



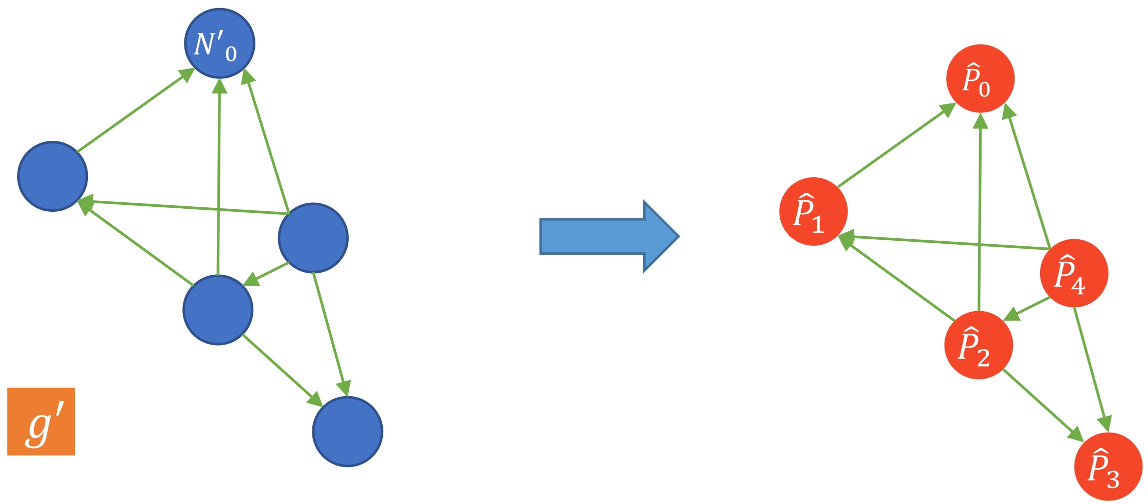
Graph Dense Layer

$\hat{P}_0 = f(N'_0; \theta_5)$

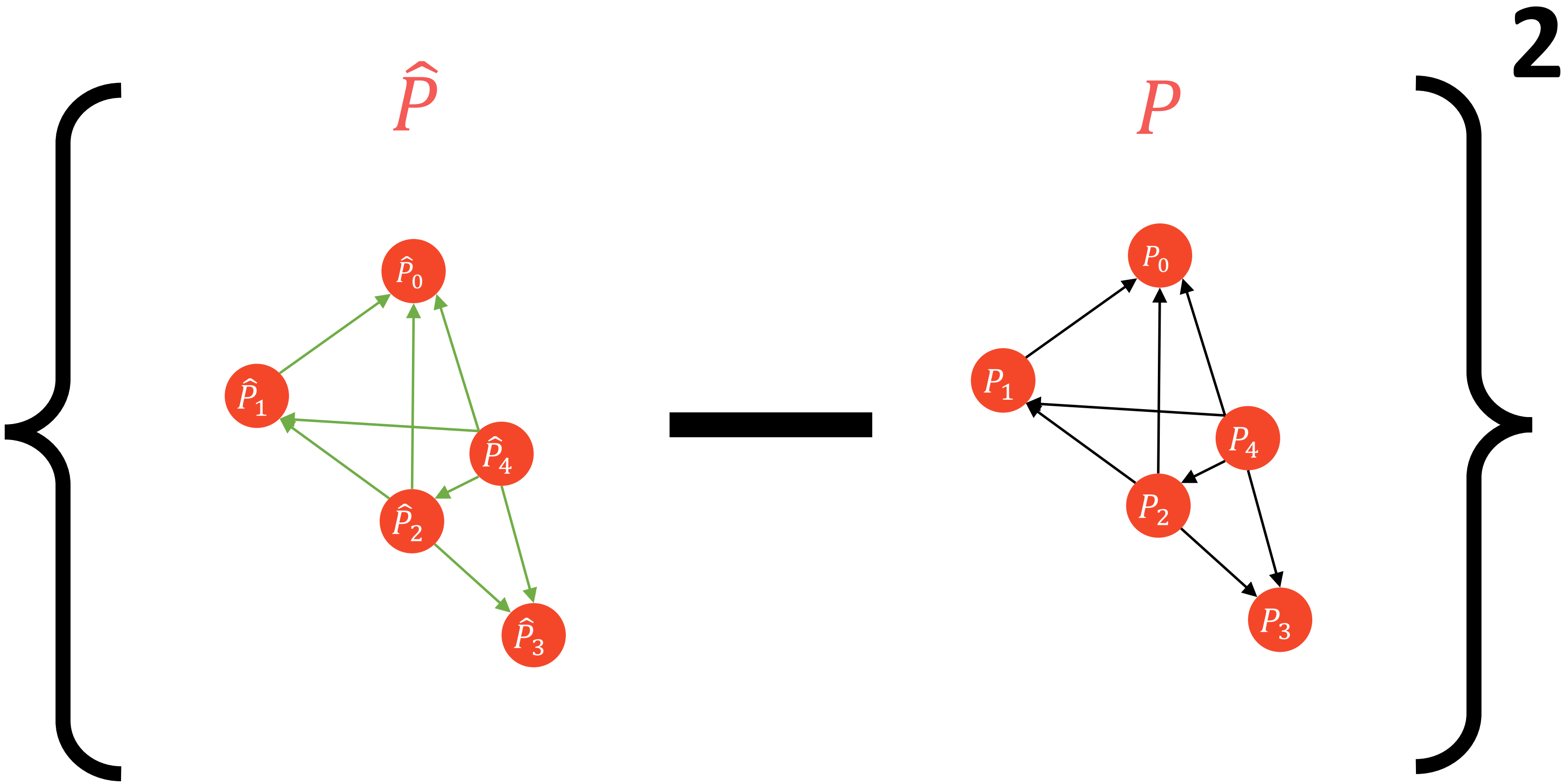


Graph Dense Layer

$\hat{P}_0 = f(N'_0; \theta_5)$

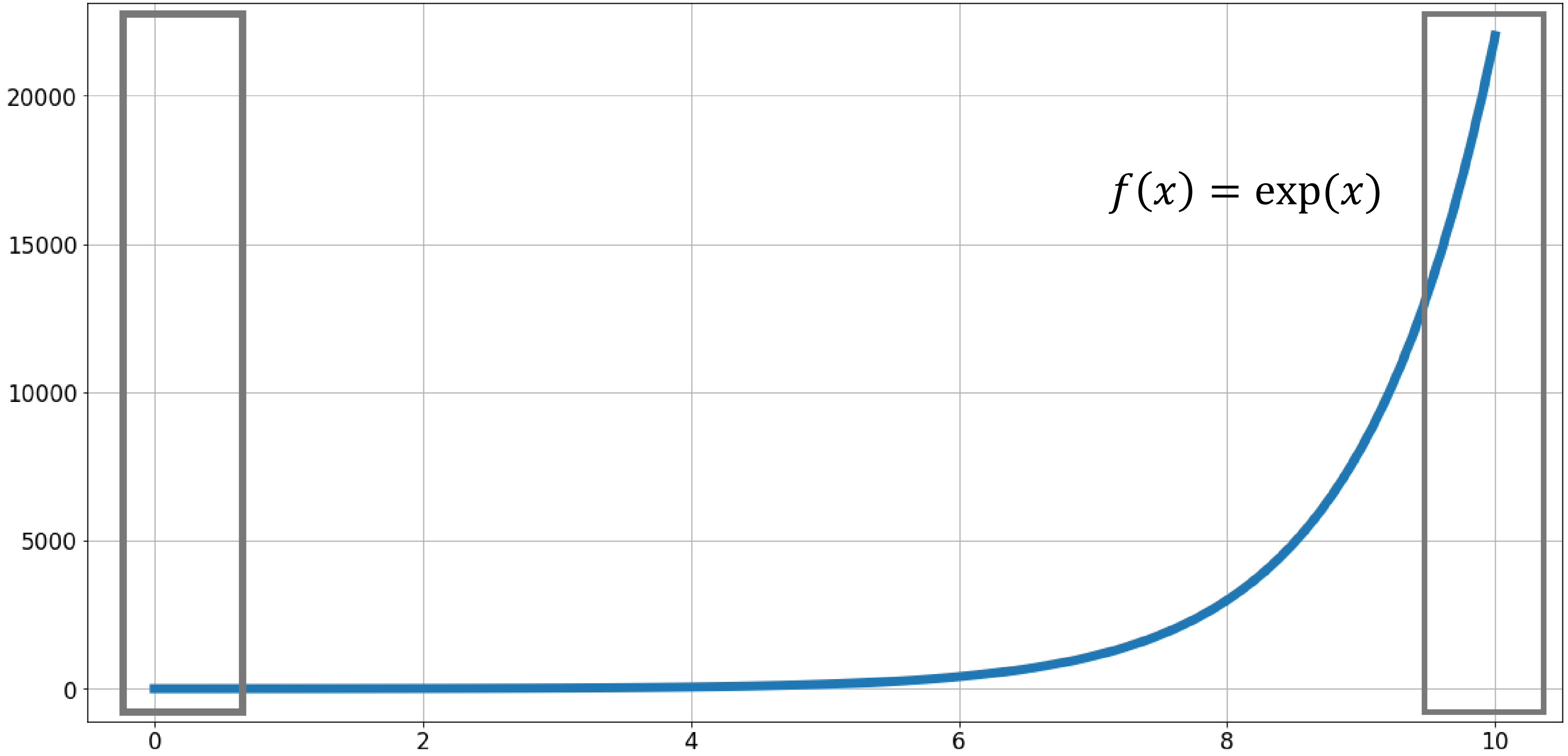


How to train your PGNN



We use mean-squared-error as a loss function of PGNN

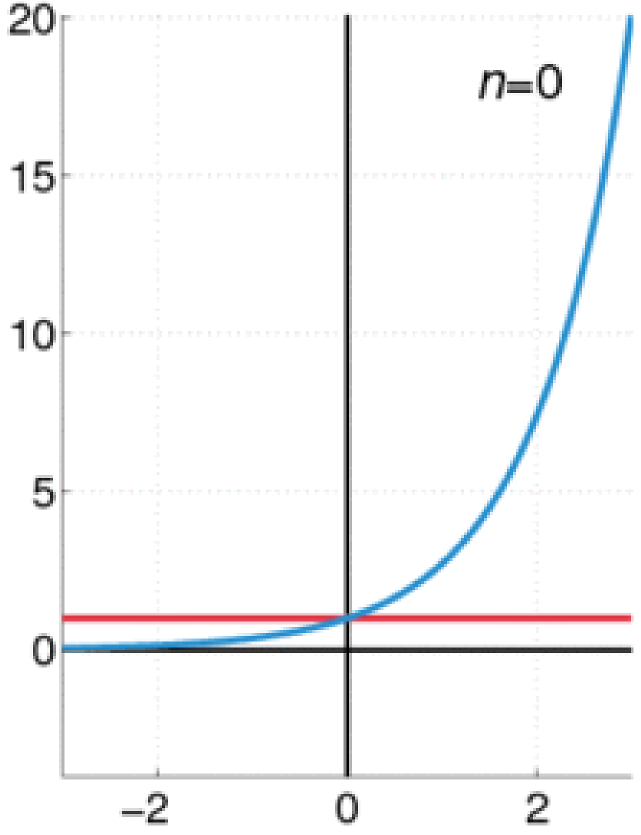
Lovely but Dreadful Exponential functions



Numerical under-flow

Numerical over-flow

Simple approximation for exponential functions

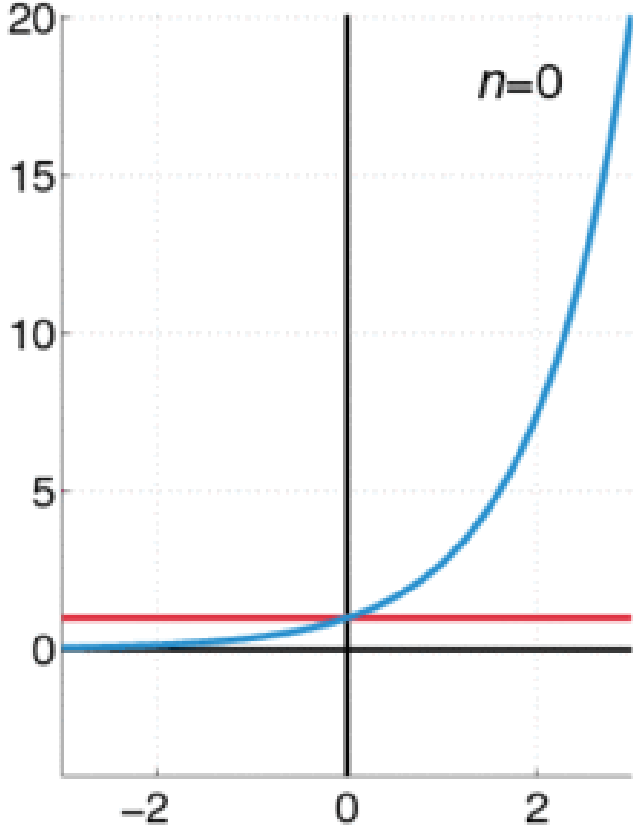


$$\exp(x) := \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

$$\approx \sum_{k=0}^{\mathrm{D}} \frac{x^k}{k!}$$

We set $\mathrm{D} = 5$

Bottom side of power-series approximation



The suggested approximation works (relatively) properly when x is small.

Question?
“why don’t you use Taylor's expansion?”
Answer:
“You may encounter exponential again!”

Scale-only normalization

Instead of using raw down stream distance d and radial wake distance r as inputs,

$$d' = \frac{d}{\sigma(d)} \times \max(0, s_d) \qquad r' = \frac{d}{\sigma(r)} \times \max(0, s_r)$$

s_d, s_r are learnable parameters

Dissect Scale-only normalization

Instead of using raw down stream distance d and radial wake distance r as inputs,

(1) (2)

$$d' = \frac{d}{\sigma(d)} \times \max(0, s_d)$$

(4)

(3)

(1) Why did you not subtract means?

→ We want the scaled values to be positive

(2) What are $\max(0, s)$ for?

→ Since s's are learnable parameters, w/o $\max(0, s)$ d' could be negative

(3) How do you get $\sigma(\cdot)$?

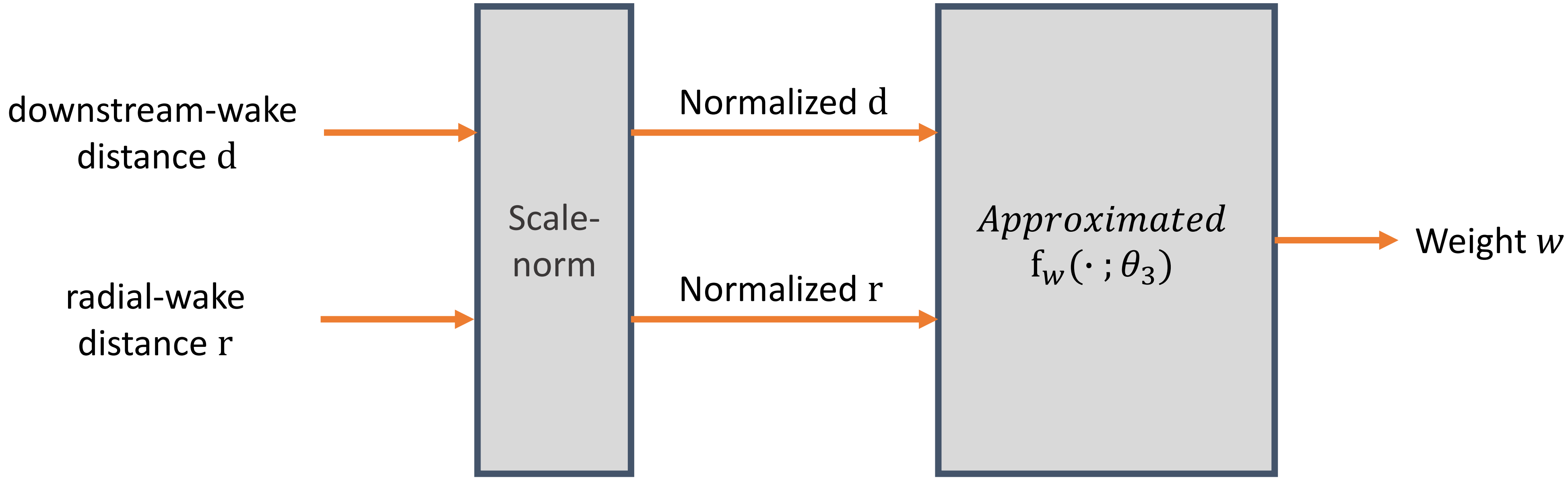
→ We employed EWMA to get $\mu(\cdot)$, $\sigma(\cdot)$ estimation

(4) Why do you multiply $\max(0, s)$ again?

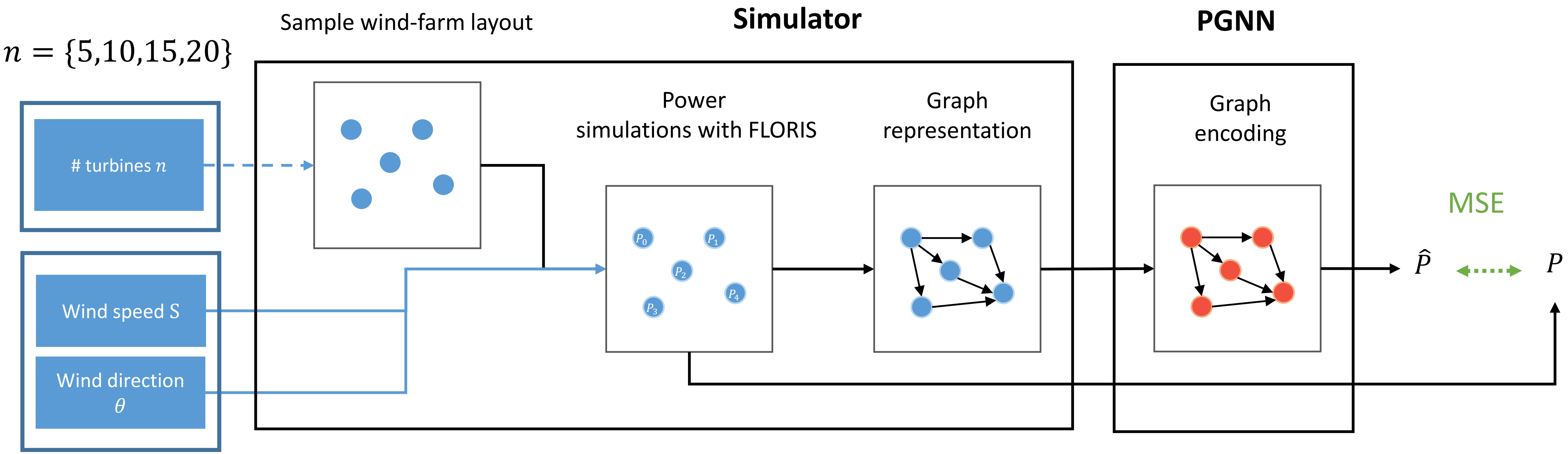
→ If 'not scaling' was the best, then we let the scaling method recover the original values d.

Same intuition Batch Normalization did.

Approximated weighting function

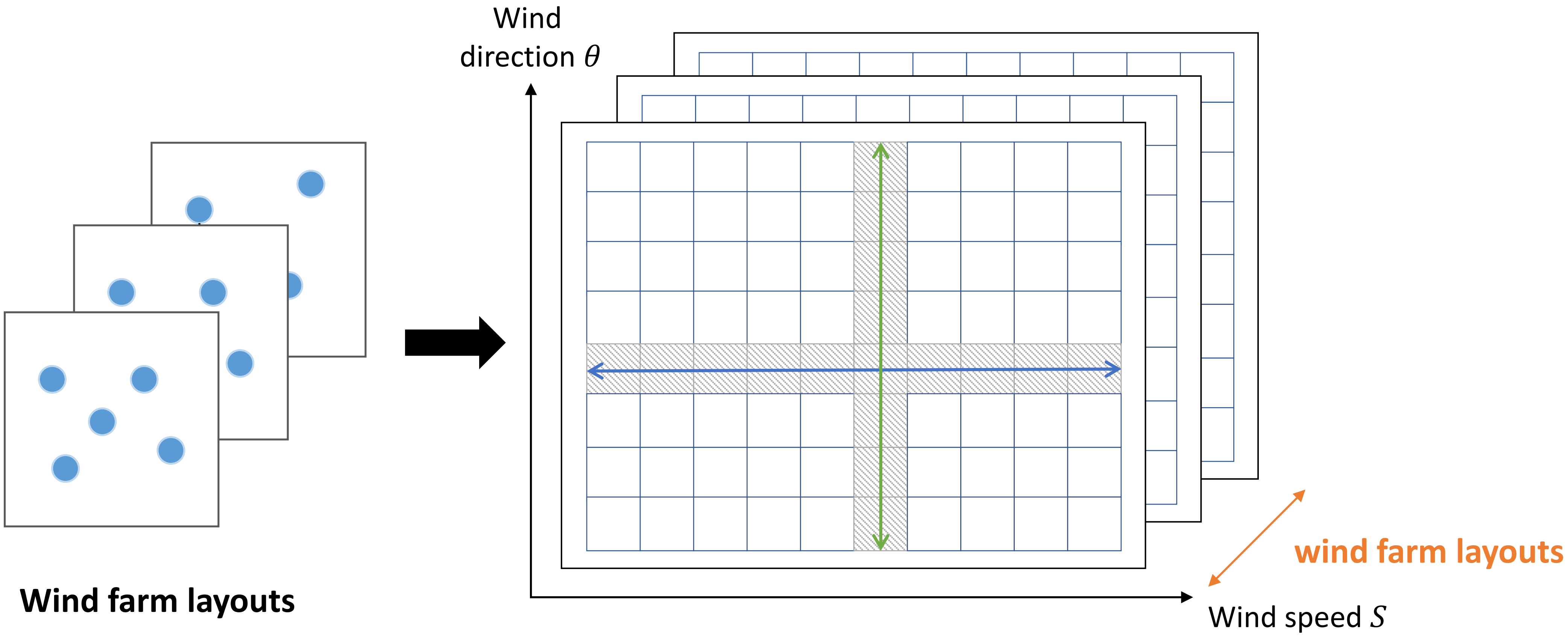


Training Procedure



sample $s \sim U(5.0m/s, 15.0m/s), \theta \sim U(0°, 360°)$

Generalization Tests

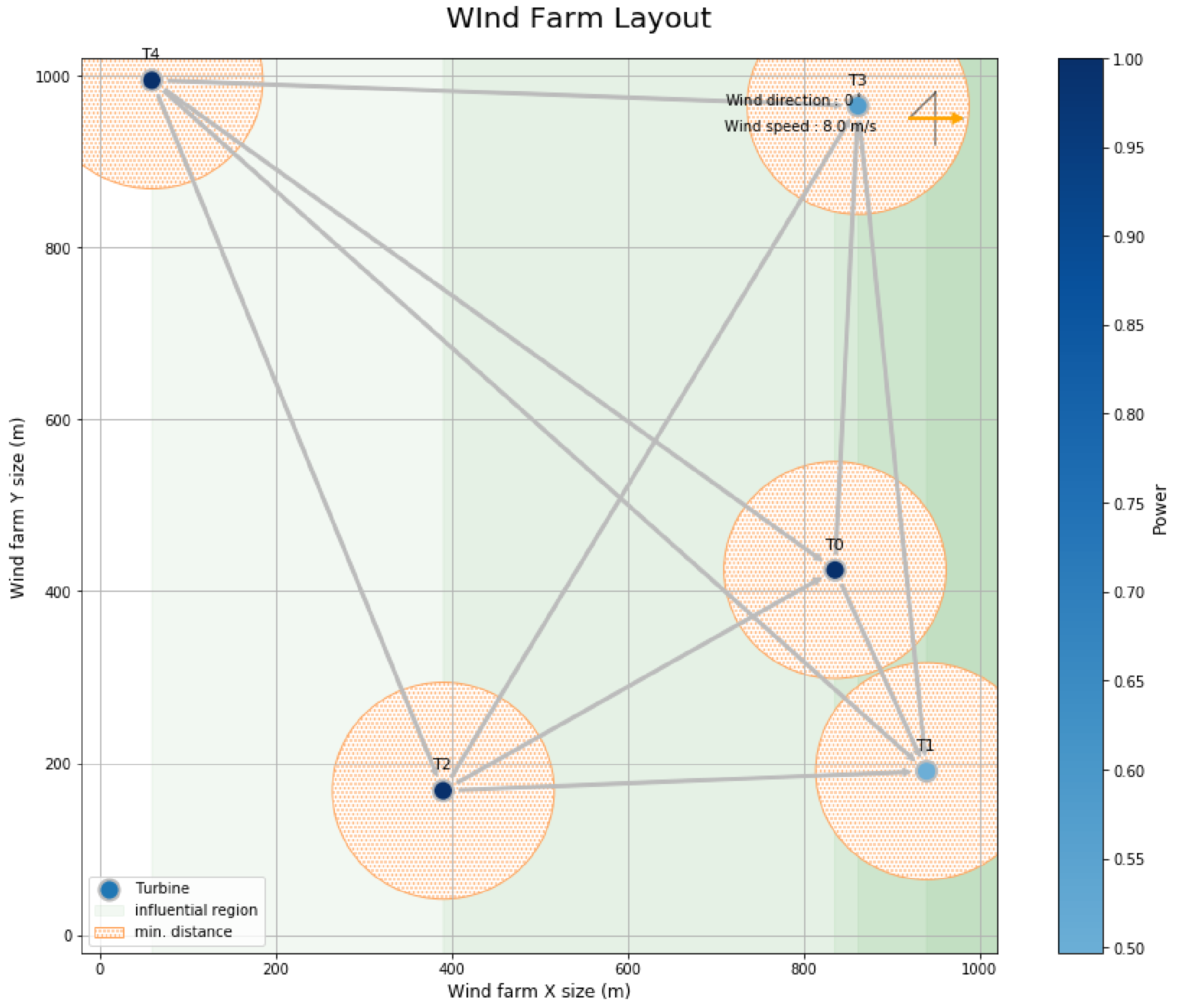


Generalization over environmental factors

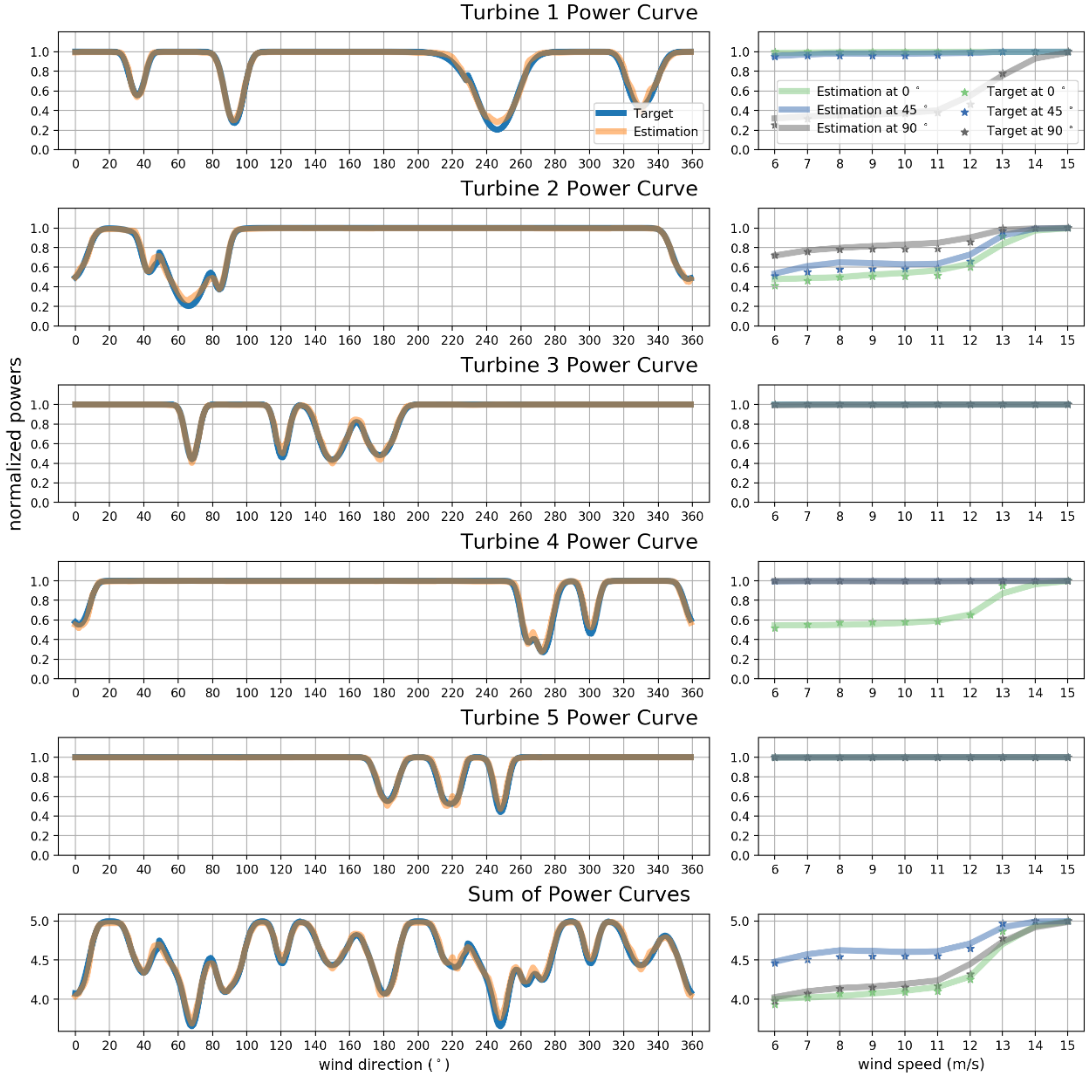
- **wind directions**, **wind speeds**

Generalization over wind farm layouts

Generalization Over Environmental Factors



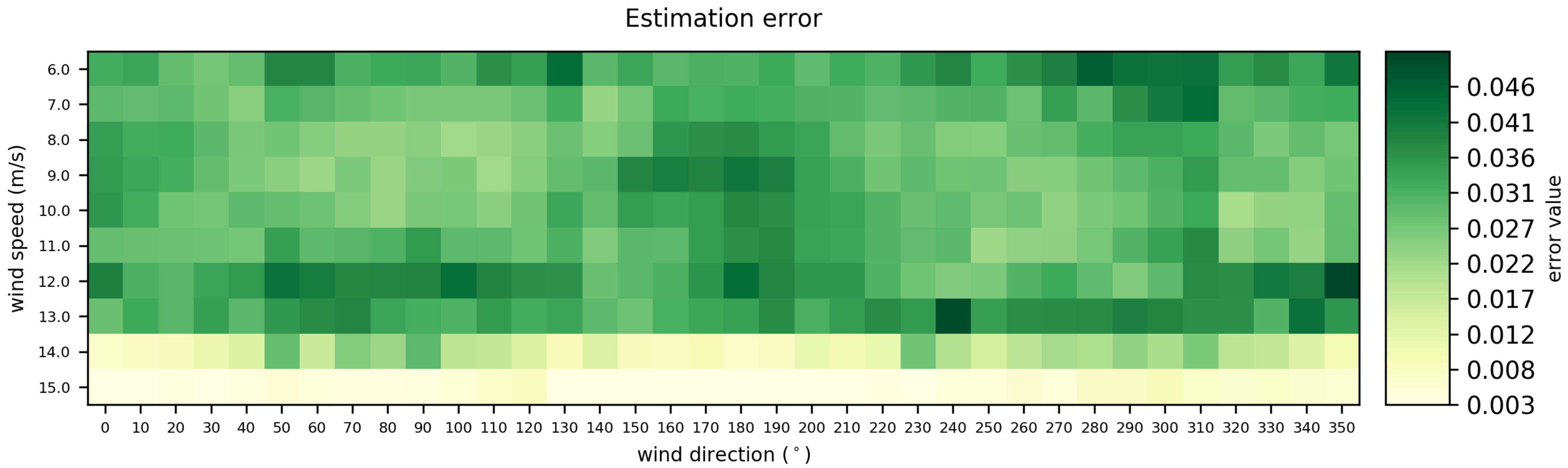
Wind speed = 8.0 m/s



Error = 0.0172

Error = 0.022

Generalization Over Layouts

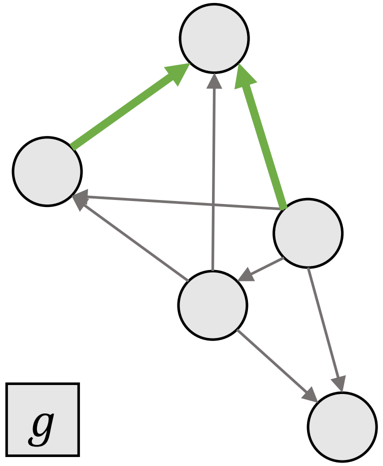


- We sample *__20 wind farm layouts,__* and estimate average estimation errors.
- Each layout has 20 wind turbines in it.

Qualitative Analysis on Physics-induced Bias

$W_{4,1} = f(inputs;\theta_3)$

$Edge'_{4,1} = W_{4,1} \times f(Edge_{4,1}\ Node_4, Node_1, g;\theta_0)$



$f(inputs;\theta_3)$

Data-induced Bias

Physics-induced Bias

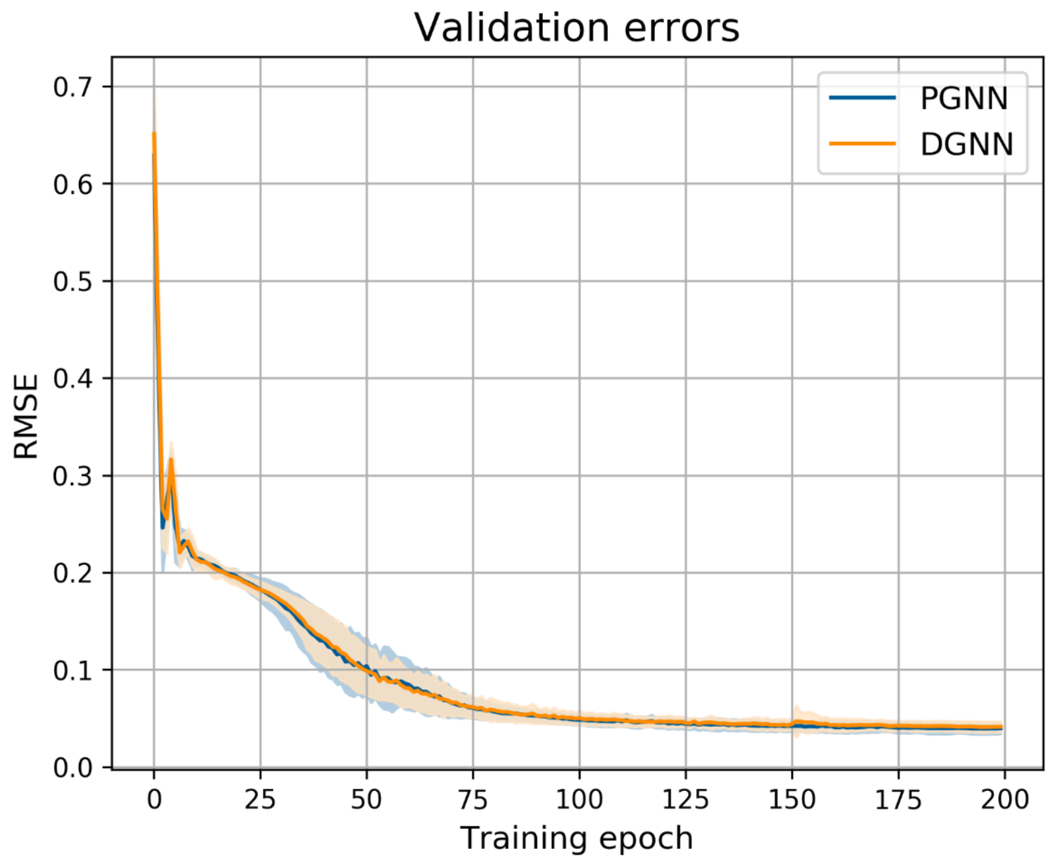
DGNN

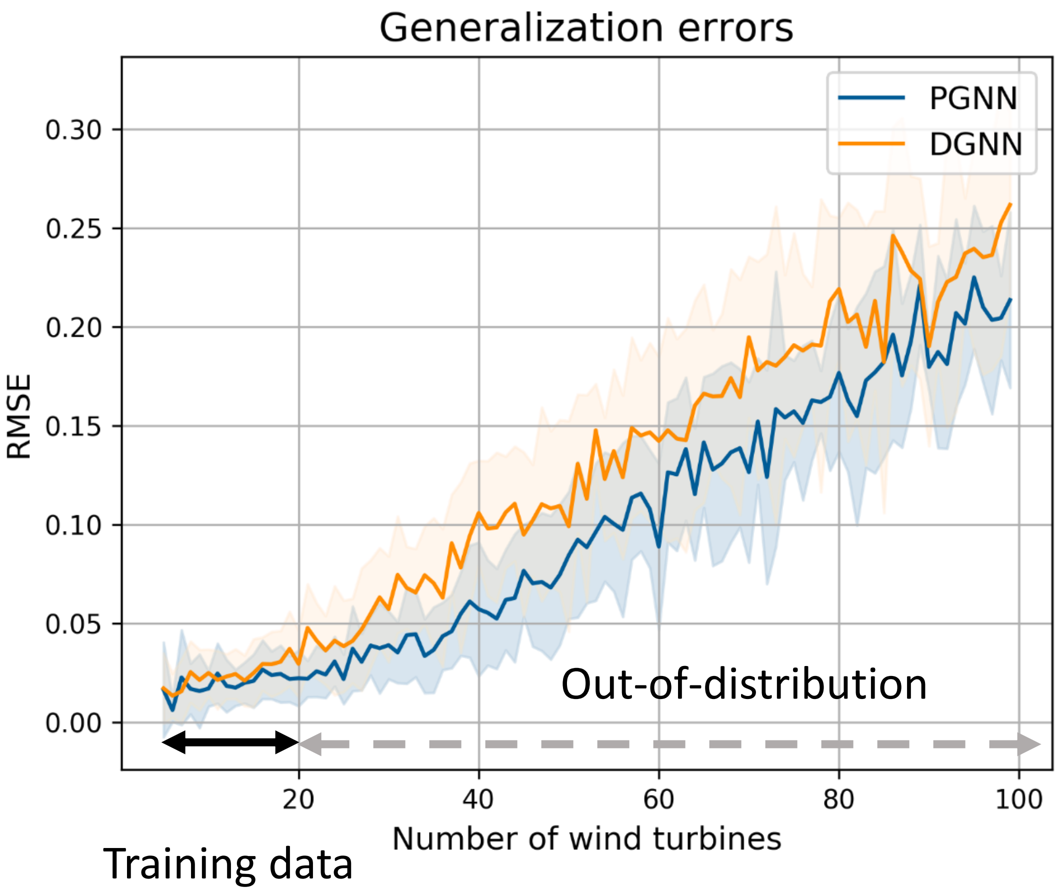
f is another neural network

PGNN

$$f(r,d;\alpha,\kappa,R_0) = 2\alpha\left(\frac{R_0}{R_0+\kappa d}\right)^2 \exp\left(-\left(\frac{r}{R_0+\kappa d}\right)^2\right)$$

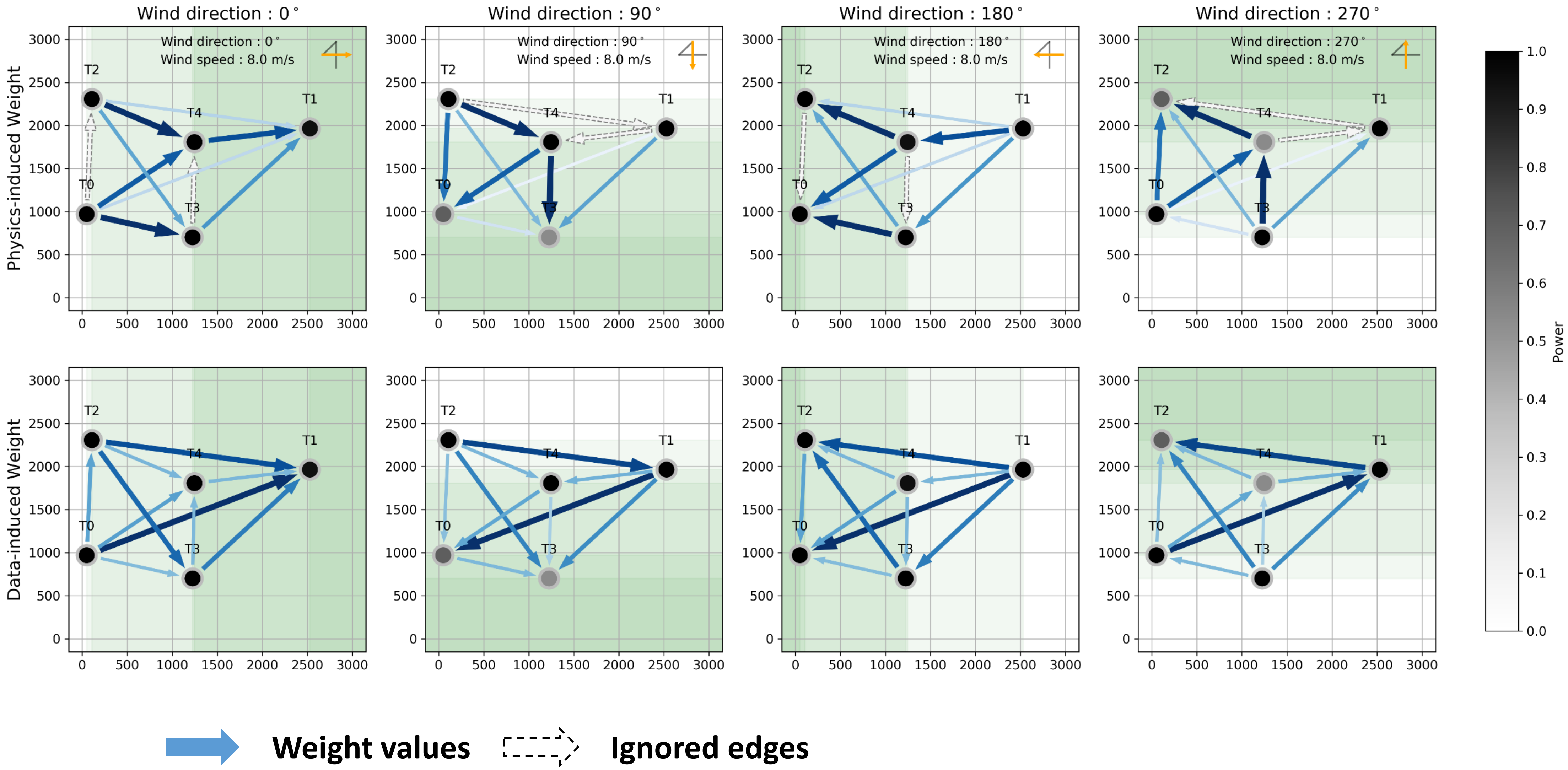
Qualitative Analysis on Physics-induced Bias



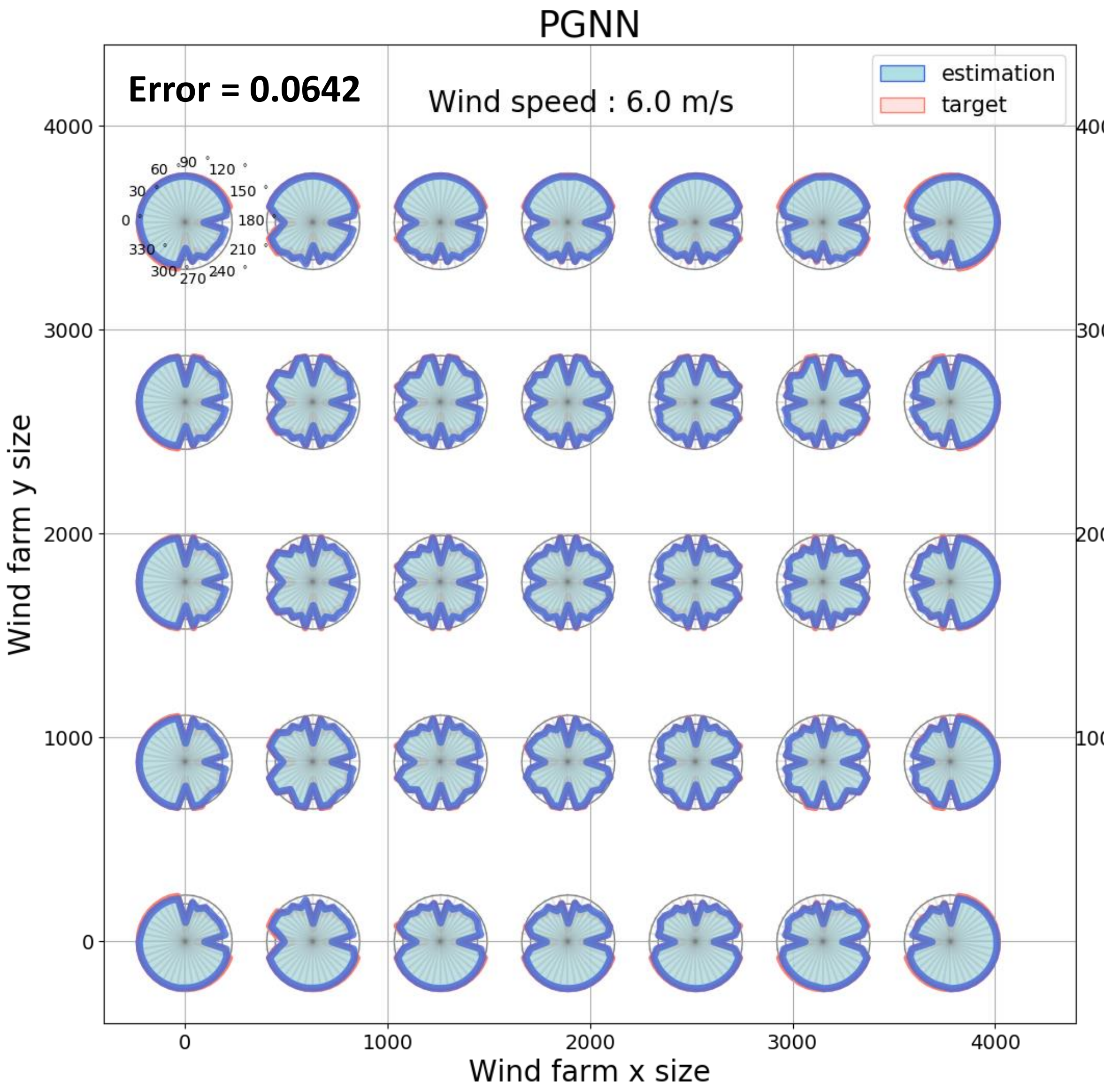


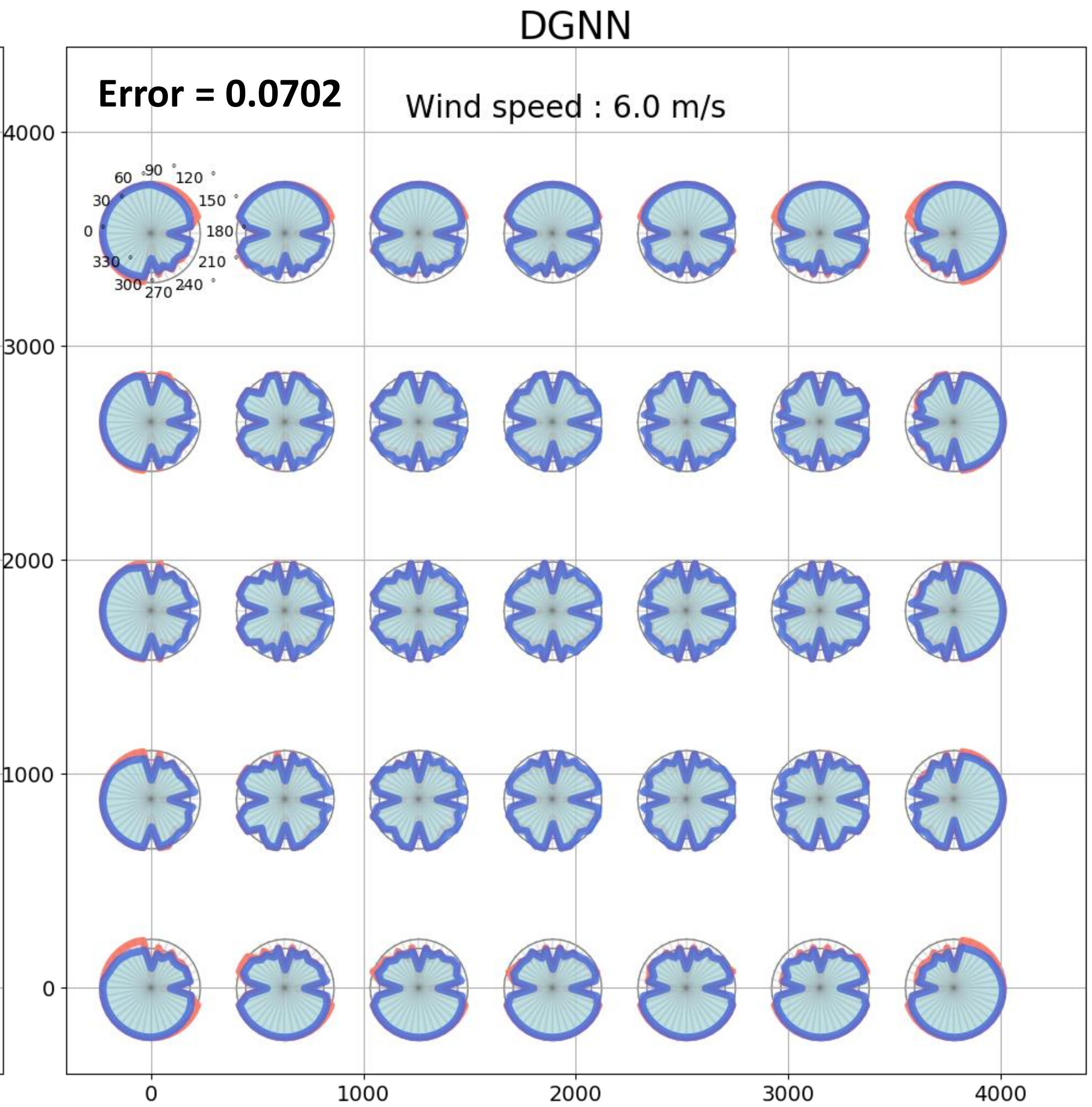
PGNN achieved 11% smaller validation error than DGNN

Case Study on Inferred Weights



Case Study on a Regularized Grid Layout





Anyway the wind blows

Junyoung Park
SYSTEMS INTELLIGENCE Lab
Industrial and Systems Engineering (ISysE)
KAIST

<https://wall.alphacoders.com/big.php?i=526859>

Normalizing powers