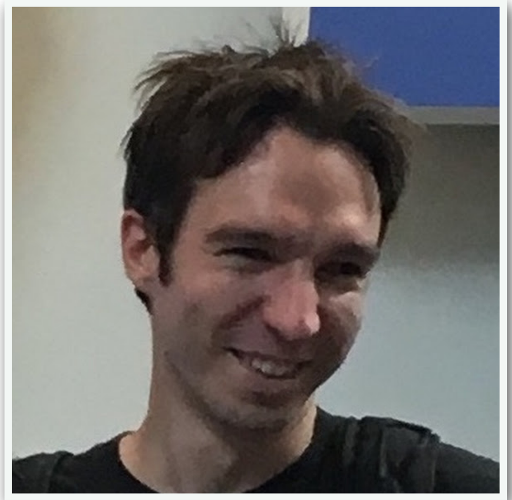
Fast Finite Width (Neural) Tangent Kernel (NTK)

Stop the war
Stand with Ukraine

ICML 2022

$$\frac{\partial f(\theta, x_1)}{\partial \theta} \frac{\partial f(\theta, x_2)}{\partial \theta}^T$$



[Roman Novak](#)

[Jascha Sohl-Dickstein](#)

[Samuel S. Shoenholz](#)

June 2022

Presentation by Roman Novak

[github.com/google/neural-tangents](#)

Plan

1. **NTK definition and notation**
2. Recap of Automatic Differentiation (AD)
3. Baseline NTK computation complexity
4. Two new algorithms for computing the NTK
5. Benchmarks

Stop the war
Stand with Ukraine

Definition

Neural network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$ Parameters $\theta \in \mathbb{R}^{\mathbf{P}}$

Stop the war
Stand with Ukraine

Definition

Neural network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$ Parameters $\theta \in \mathbb{R}^{\mathbf{P}}$

The NTK is the outer product of Jacobians:

$$\Theta_\theta(x_1, x_2) := \frac{\partial f(\theta, x_1)}{\partial \theta} \frac{\partial f(\theta, x_2)}{\partial \theta}^T$$

Stop the war
Stand with Ukraine

Definition

Stop the war
Stand with Ukraine

Neural network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$ Parameters $\theta \in \mathbb{R}^{\mathbf{P}}$

The NTK is the outer product of Jacobians:

$$\underbrace{\Theta_\theta(x_1, x_2)}_{\mathbf{O}\times\mathbf{O}} := \underbrace{\frac{\partial f(\theta, x_1)}{\partial \theta}}_{\mathbf{O}\times\mathbf{P}} \underbrace{\frac{\partial f(\theta, x_2)}{\partial \theta}^T}_{\mathbf{P}\times\mathbf{O}}$$

Stop the war
Stand with Ukraine

Definition

Neural network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$ Parameters $\theta \in \mathbb{R}^{\mathbf{P}}$

Batch size $\mathbf{N}$

The NTK is the outer product of Jacobians:

$$\underbrace{\Theta_\theta(x_1, x_2)}_{\mathbf{NO}\times\mathbf{NO}} = \underbrace{\frac{\partial f(\theta, x_1)}{\partial \theta}}_{\mathbf{NO}\times\mathbf{P}} \underbrace{\frac{\partial f(\theta, x_2)}{\partial \theta}^T}_{\mathbf{P}\times\mathbf{NO}}$$

Stop the war
Stand with Ukraine

Definition

Neural network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$ Parameters $\theta \in \mathbb{R}^{\mathbf{P}}$

Batch size $\mathbf{N}$

The NTK is the outer product of Jacobians:

$$\underbrace{\Theta_\theta(x_1, x_2)}_{\mathbf{NO} \times \mathbf{NO}} = \underbrace{\frac{\partial f(\theta, x_1)}{\partial \theta}}_{\mathbf{NO} \times \mathbf{P}} \underbrace{\frac{\partial f(\theta, x_2)}{\partial \theta}^T}_{\mathbf{P} \times \mathbf{NO}}$$

Applications in approximate inference, deep learning theory, NAS, MAML, etc.

Definition

Stop the war
Stand with Ukraine

Neural network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$ Parameters $\theta \in \mathbb{R}^{\mathbf{P}}$ Batch size $\mathbf{N}$

The NTK is the outer product of Jacobians:

$$\underbrace{\Theta_\theta(x_1, x_2)}_{\mathbf{NO}\times\mathbf{NO}} = \underbrace{\frac{\partial f(\theta, x_1)}{\partial \theta}}_{\mathbf{NO}\times\mathbf{P}} \underbrace{\frac{\partial f(\theta, x_2)}{\partial \theta}^T}_{\mathbf{P}\times\mathbf{NO}}$$

Very costly contraction: $\mathbf{N}^2\mathbf{O}^2\mathbf{P}$ time!

Stop the war
Stand with Ukraine

Definition

Neural network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$ Parameters $\theta \in \mathbb{R}^{\mathbf{P}}$ Batch size $\mathbf{N}$

The NTK is the outer product of Jacobians:

1000^2 for ImageNet

$$\underbrace{\Theta_\theta(x_1, x_2)}_{\mathbf{NO}\times\mathbf{NO}} = \underbrace{\frac{\partial f(\theta, x_1)}{\partial \theta}}_{\mathbf{NO}\times\mathbf{P}} \underbrace{\frac{\partial f(\theta, x_2)}{\partial \theta}^T}_{\mathbf{P}\times\mathbf{NO}}$$

Tens of millions

Very costly contraction: $\mathbf{N}^2\mathbf{O}^2\mathbf{P}$ time!

Stop the war
Stand with Ukraine

Definition

Neural network $f\left(\theta, x\right) \in \mathbb{R}^{\mathbf{O}}$ Parameters $\theta \in \mathbb{R}^{\mathbf{P}}$ Batch size $\mathbf{N}$

The NTK is the outer product of Jacobians:

$$\underbrace{\Theta_\theta(x_1, x_2)}_{\mathbf{NO}\times\mathbf{NO}} = \underbrace{\frac{\partial f(\theta, x_1)}{\partial \theta}}_{\mathbf{NO}\times\mathbf{P}} \underbrace{\frac{\partial f(\theta, x_2)}{\partial \theta}^T}_{\mathbf{P}\times\mathbf{NO}}$$

This work: reducing the contraction cost.

Very costly contraction: ~~$\mathbf{N}^2\mathbf{O}^2\mathbf{P}$~~ time!

Plan

1. NTK definition and notation
2. **Recap of Automatic Differentiation (AD)**
3. Baseline NTK computation complexity
4. Two new algorithms for computing the NTK
5. Benchmarks

Stop the war
Stand with Ukraine

Recap of Automatic Differentiation (AD)

Stop the war
Stand with Ukraine

Two fundamental AD building blocks:

- **Jacobian-vector product** $\mathrm{JVP}_{(f,\theta,x)} : \theta_t \in \mathbb{R}^{\mathbf{P}} \mapsto \frac{\partial f(\theta, x)}{\partial \theta} \theta_t \in \mathbb{R}^{\mathbf{O}}$

- **vector-Jacobian product** $\mathrm{VJP}_{(f,\theta,x)} : f_c \in \mathbb{R}^{\mathbf{O}} \mapsto \frac{\partial f(\theta, x)}{\partial \theta}^T f_c \in \mathbb{R}^{\mathbf{P}}$

Neural Network	Output size	Parameters	Batch size
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$

Recap of Automatic Differentiation (AD)

Stop the war
Stand with Ukraine

Every other AD operation is implemented with them:

- Gradient $\texttt{grad}(f)(\theta, x) = \texttt{VJP}_{f,\theta,x}(1)$
- Reverse-mode Jacobian $\texttt{jacrev}(f)(\theta, x) = \texttt{vmap}(\texttt{VJP}_{f,\theta,x})(I_{\mathbf{O}})$
- Forward-mode Jacobian $\texttt{jacfwd}(f)(\theta, x) = \texttt{vmap}(\texttt{JVP}_{f,\theta,x})(I_{\mathbf{P}})$
- Hessian $\texttt{hessian}(f)(\theta, x) = \texttt{jacfwd}(\texttt{jacrev}(f)(\theta, x))(\theta, x)$
- ...

Neural Network	Output size	Parameters	Batch size
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$

Stop the war
Stand with Ukraine

Recap of Automatic Differentiation (AD)

- **Time complexities of the forward pass (FP), JVP and VJP:**

Op	**Forward Pass (FP)**	**Jacobian-vector Product (JVP)**	**Vector-Jacobian Product (VJP)**
Time	$\mathbf{N}\,[\mathbf{FP}]$		

Neural Network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$

Output size $\mathbf{O}$

Parameters $\theta \in \mathbb{R}^{\mathbf{P}}$

Batch size $\mathbf{N}$

Forward pass $[\mathbf{FP}]$

Recap of Automatic Differentiation (AD)

Stop the war
Stand with Ukraine

- **Time complexities of the reverse-mode Jacobian:**

$$\underbrace{\frac{\partial f(\theta, x)^T}{\partial \theta}}_{\mathbf{P} \times \mathbf{O}} := \frac{\partial f(\theta, x)}{\partial \theta}^T I_{\mathbf{O}} = \underbrace{\left[\mathrm{VJP}_{(f,\theta,x)}\left(e_1\right), \ldots, \mathrm{VJP}_{(f,\theta,x)}\left(e_{\mathbf{O}}\right)\right]}_{\mathbf{O}\ \mathrm{VJPs}}$$

Neural Network	Output size	Parameters	Batch size	Forward pass
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$	$[\mathbf{FP}]$

Stop the war
Stand with Ukraine

Recap of Automatic Differentiation (AD)

Given Jacobian definitions, we can compute its cost via JVP/VJP costs.

Op	Forward Pass (FP)	Jacobian-vector Product (JVP)	Vector-Jacobian Product (VJP)	Jacobian (reverse-mode)
Time	$\mathbf{N}\,[\mathbf{FP}]$			$\mathbf{N}\mathbf{O}\,[\mathbf{FP}]$

Neural Network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$

Output size $\mathbf{O}$

Parameters $\theta \in \mathbb{R}^{\mathbf{P}}$

Batch size $\mathbf{N}$

Forward pass $[\mathbf{FP}]$

Plan

1. NTK definition and notation
2. Recap of Automatic Differentiation (AD)
3. **Baseline NTK computation complexity**
4. Two new algorithms for computing the NTK
5. Benchmarks

Stop the war
Stand with Ukraine

Baseline: Jacobian contraction

Stop the war
Stand with Ukraine

Recall that we want to compute

$$\underbrace{\Theta_\theta(x_1, x_2)}_{\mathbf{NO}\times\mathbf{NO}} = \underbrace{\frac{\partial f(\theta, x_1)}{\partial \theta}}_{\mathbf{NO}\times\mathbf{P}} \underbrace{\frac{\partial f(\theta, x_2)}{\partial \theta}^T}_{\mathbf{P}\times\mathbf{NO}}$$

Neural Network	Output size	Parameters	Batch size	Forward pass
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$	$[\mathbf{FP}]$

Baseline: Jacobian contraction

Stand with Ukraine
Stop the war

Baseline algorithm: compute two Jacobians and contract them.

$$\underbrace{\Theta_\theta(x_1, x_2)}_{\mathbf{NO}\times\mathbf{NO}} = \underbrace{\frac{\partial f(\theta, x_1)}{\partial \theta}}_{\mathbf{NO}\times\mathbf{P}} \underbrace{\frac{\partial f(\theta, x_2)}{\partial \theta}^T}_{\mathbf{P}\times\mathbf{NO}}$$

Reverse-mode Jacobian + contraction

Neural Network	Output size	Parameters	Batch size	Forward pass
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$	$[\mathbf{FP}]$

Stop the war
Stand with Ukraine

Baseline: Jacobian contraction

Baseline algorithm: compute two Jacobians and contract them.

$$\underbrace{\Theta_\theta(x_1, x_2)}_{\mathbf{NO} \times \mathbf{NO}} = \underbrace{\frac{\partial f(\theta, x_1)}{\partial \theta}}_{\mathbf{NO} \times \mathbf{P}} \underbrace{\frac{\partial f(\theta, x_2)}{\partial \theta}^T}_{\mathbf{P} \times \mathbf{NO}}$$

Reverse-mode Jacobian + $\mathbf{N}^2\mathbf{O}^2\mathbf{P}$

Neural Network	Output size	Parameters	Batch size	Forward pass
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$	$[\mathbf{FP}]$

Stop the war
Stand with Ukraine

Baseline: Jacobian contraction

Baseline algorithm: compute two Jacobians and contract them.

$$\underbrace{\Theta_\theta(x_1, x_2)}_{\mathbf{NO}\times\mathbf{NO}} = \underbrace{\frac{\partial f(\theta, x_1)}{\partial \theta}}_{\mathbf{NO}\times\mathbf{P}} \underbrace{\frac{\partial f(\theta, x_2)}{\partial \theta}^T}_{\mathbf{P}\times\mathbf{NO}}$$

$$\mathbf{NO}\,[\mathbf{FP}] + \mathbf{NO}\,[\mathbf{FP}] + \mathbf{N}^2\mathbf{O}^2\mathbf{P}$$

Neural Network	Output size	Parameters	Batch size	Forward pass
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$	$[\mathbf{FP}]$

Baseline: Jacobian contraction

Method	Jacobian Contraction
Time	$\mathbf{N}\mathbf{O}\,[\mathbf{FP}] + \mathbf{N}^2\mathbf{O}^2\mathbf{P}$

Neural Network

$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$

Output size

$\mathbf{O}$

Parameters

$\theta \in \mathbb{R}^{\mathbf{P}}$

Batch size

$\mathbf{N}$

Forward pass

$[\mathbf{FP}]$

Stop the war
Stand with Ukraine

Plan

1. NTK definition and notation
2. Recap of Automatic Differentiation (AD)
3. Baseline NTK computation complexity
4. **Two new algorithms for computing the NTK**
5. Benchmarks

Stop the war
Stand with Ukraine

Idea 1: NTK-vector products

Similarly to Jacobians, compute the NTK via NTK-vector products applied to $\mathbf{O}$ columns of the identity matrix $I_{\mathbf{O}}$.

Neural Network	Output size	Parameters	Batch size	Forward pass
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$	$[\mathbf{FP}]$

Stop the war
Stand with Ukraine

Idea 1: NTK-vector products

Similarly to Jacobians, compute the NTK via NTK-vector products applied to $\mathbf{O}$ columns of the identity matrix $I_{\mathbf{O}}$. For $\mathbf{N} = 1$:

Evaluate $\Theta_\theta(x_1, x_2) = \Theta_\theta(x_1, x_2) I_{\mathbf{O}}$ row-by-row.

Neural Network	Output size	Parameters	Batch size	Forward pass
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$	$[\mathbf{FP}]$

Stop the war
Stand with Ukraine

Stop the war
Stand with Ukraine

Idea 1: NTK-vector products

Similarly to Jacobians, compute the NTK via NTK-vector products applied to $\mathbf{O}$ columns of the identity matrix $I_{\mathbf{O}}$. For $\mathbf{N} = 1$:

Evaluate $\Theta_\theta(x_1, x_2) = \Theta_\theta(x_1, x_2) I_{\mathbf{O}}$ row-by-row.

For a single input column v:

$$\Theta_\theta(x_1, x_2)v$$

Neural Network	Output size	Parameters	Batch size	Forward pass
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$	$[\mathbf{FP}]$

Stop the war
Stand with Ukraine

Idea 1: NTK-vector products

Similarly to Jacobians, compute the NTK via NTK-vector products applied to $\mathbf{O}$ columns of the identity matrix $I_{\mathbf{O}}$. For $\mathbf{N} = 1$:

Evaluate $\Theta_\theta(x_1, x_2) = \Theta_\theta(x_1, x_2) I_{\mathbf{O}}$ row-by-row.

For a single input column v:

$$\Theta_\theta(x_1, x_2) v = \frac{\partial f(\theta, x_1)}{\partial \theta} \frac{\partial f(\theta, x_2)}{\partial \theta}^T v$$

Neural Network	Output size	Parameters	Batch size	Forward pass
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$	$[\mathbf{FP}]$

Idea 1: NTK-vector products

Similarly to Jacobians, compute the NTK via NTK-vector products applied to $\mathbf{O}$ columns of the identity matrix $I_{\mathbf{O}}$. For $\mathbf{N} = 1$:

Evaluate $\Theta_\theta(x_1, x_2) = \Theta_\theta(x_1, x_2) I_{\mathbf{O}}$ row-by-row.

For a single input column v:

$$\Theta_\theta(x_1, x_2)v = \frac{\partial f\left(\theta, x_1\right)}{\partial \theta} \frac{\partial f\left(\theta, x_2\right)}{\partial \theta}^T v = \frac{\partial f\left(\theta, x_1\right)}{\partial \theta} \mathrm{VJP}_{(f, \theta, x_2)}\left(v\right)$$

Neural Network $f\left(\theta, x\right) \in \mathbb{R}^{\mathbf{O}}$

Output size $\mathbf{O}$

Parameters $\theta \in \mathbb{R}^{\mathbf{P}}$

Batch size $\mathbf{N}$

Forward pass $[\mathbf{FP}]$

Stop the war
Stand with Ukraine

Stand with Ukraine
Stop the war

Idea 1: NTK-vector products

Similarly to Jacobians, compute the NTK via NTK-vector products applied to $\mathbf{O}$ columns of the identity matrix $I_{\mathbf{O}}$. For $\mathbf{N} = 1$:

Evaluate $\Theta_\theta(x_1, x_2) = \Theta_\theta(x_1, x_2) I_{\mathbf{O}}$ row-by-row.

For a single input column v:

$$\Theta_\theta(x_1, x_2)v = \frac{\partial f(\theta, x_1)}{\partial \theta} \frac{\partial f(\theta, x_2)}{\partial \theta}^T v = \frac{\partial f(\theta, x_1)}{\partial \theta} \mathrm{VJP}_{(f,\theta,x_2)}(v) = \mathrm{JVP}_{(f,\theta,x_1)}\left[\mathrm{VJP}_{(f,\theta,x_2)}(v)\right]$$

Neural Network	Output size	Parameters	Batch size	Forward pass
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$	$[\mathbf{FP}]$

Stand with Ukraine
Stop the war

Idea 1: NTK-vector products

Similarly to Jacobians, compute the NTK via NTK-vector products applied to $\mathbf{O}$ columns of the identity matrix $I_{\mathbf{O}}$. For $\mathbf{N} = 1$:

Evaluate $\Theta_\theta(x_1, x_2) = \Theta_\theta(x_1, x_2) I_{\mathbf{O}}$ row-by-row.

For a single input column v:

$[\mathbf{FP}]$

$$\Theta_\theta(x_1, x_2)v = \frac{\partial f(\theta, x_1)}{\partial \theta}\frac{\partial f(\theta, x_2)}{\partial \theta}^T v = \frac{\partial f(\theta, x_1)}{\partial \theta}\mathrm{VJP}_{(f,\theta,x_2)}(v) = \boxed{\mathrm{JVP}_{(f,\theta,x_1)}\left[\mathrm{VJP}_{(f,\theta,x_2)}(v)\right]}$$

Neural Network	Output size	Parameters	Batch size	Forward pass
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$	$[\mathbf{FP}]$

Stand with Ukraine
Stop the war

Idea 1: NTK-vector products

Similarly to Jacobians, compute the NTK via NTK-vector products applied to $\mathbf{O}$ columns of the identity matrix $I_{\mathbf{O}}$. For $\mathbf{N} = 1$:

Evaluate $\Theta_\theta(x_1, x_2) = \Theta_\theta(x_1, x_2) I_{\mathbf{O}}$ row-by-row.

(repeat $\mathbf{O}$ times)

$\mathbf{O}\,[\mathbf{FP}]$

For a single input column v:

$$\Theta_\theta(x_1, x_2)v = \frac{\partial f(\theta, x_1)}{\partial \theta}\frac{\partial f(\theta, x_2)}{\partial \theta}^T v = \frac{\partial f(\theta, x_1)}{\partial \theta}\mathrm{VJP}_{(f,\theta,x_2)}(v) = \mathrm{JVP}_{(f,\theta,x_1)}\left[\mathrm{VJP}_{(f,\theta,x_2)}(v)\right]$$

Neural Network	Output size	Parameters	Batch size	Forward pass
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$	$[\mathbf{FP}]$

Stop the war
Stand with Ukraine

Idea 1: NTK-vector products

Batched setting: needs to be done for every pair of x_1 and x_2!

$$\mathbf{N}^2\mathbf{O}\,[\mathbf{FP}]$$

$$\Theta_\theta(x_1, x_2)v = \frac{\partial f(\theta, x_1)}{\partial \theta}\frac{\partial f(\theta, x_2)}{\partial \theta}^T v = \frac{\partial f(\theta, x_1)}{\partial \theta}\mathrm{VJP}_{(f,\theta,x_2)}(v) = \mathrm{JVP}_{(f,\theta,x_1)}\left[\mathrm{VJP}_{(f,\theta,x_2)}(v)\right]$$

Neural Network	Output size	Parameters	Batch size	Forward pass
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$	$[\mathbf{FP}]$

Idea 1: NTK-vector products

Method	Jacobian Contraction	NTK-vector products
Time	$\mathbf{N}\mathbf{O}\,[\mathbf{FP}] + \mathbf{N}^2\mathbf{O}^2\mathbf{P}$	$\mathbf{N}^2\mathbf{O}\,[\mathbf{FP}]$

Neural Network	Output size	Parameters	Batch size	Forward pass
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$	$[\mathbf{FP}]$

Stop the war
Stand with Ukraine

Idea 2: Structured derivatives

Exploit the structure in primitive derivatives for computing the NTK.

Neural Network	Output size	Parameters	Batch size	Forward pass
$f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$	$\mathbf{O}$	$\theta \in \mathbb{R}^{\mathbf{P}}$	$\mathbf{N}$	$[\mathbf{FP}]$

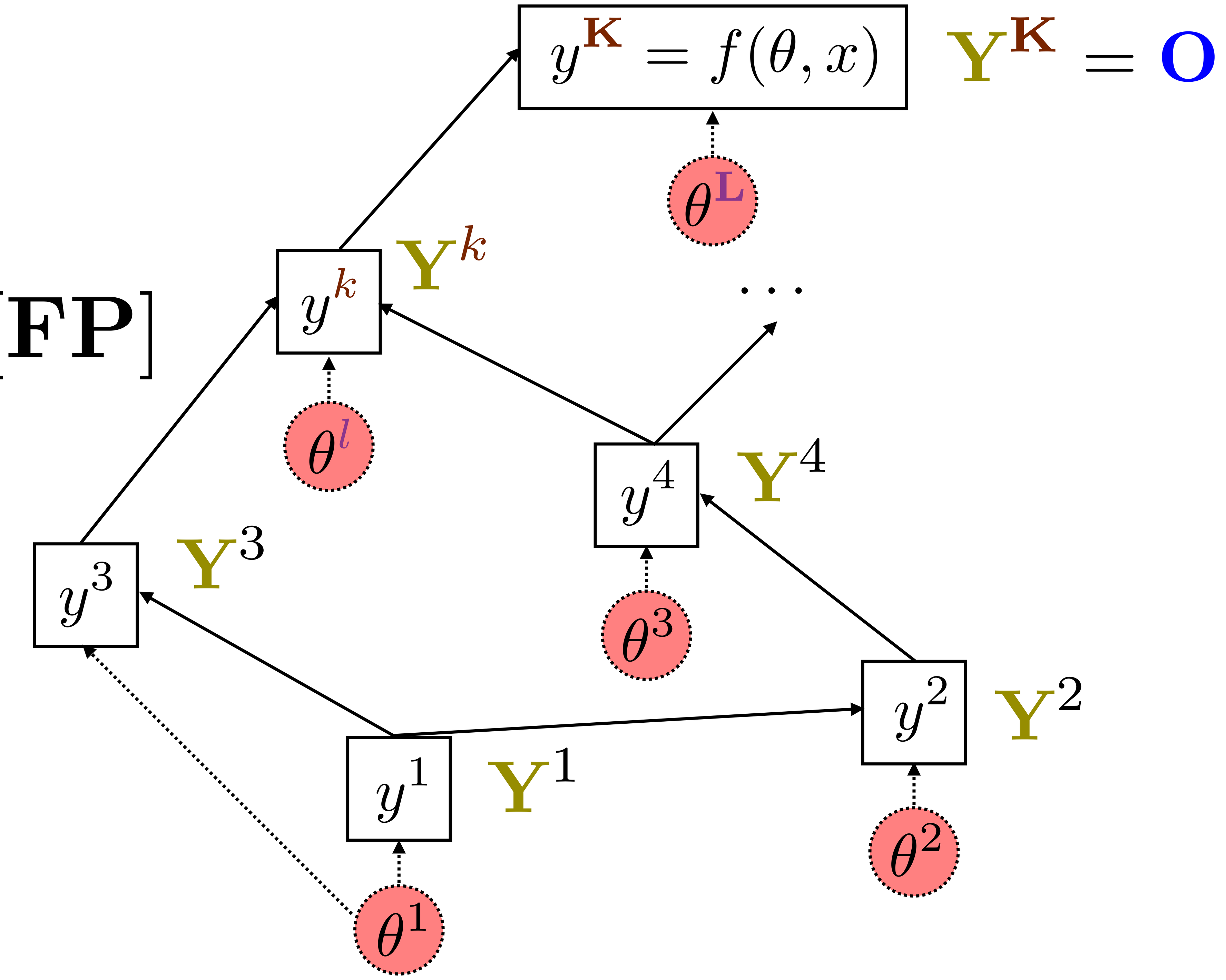
Stop the war
Stand with Ukraine

Stop the war
Stand with Ukraine

Idea 2: Structured derivatives

Consider the computation graph of the function f:

Forward Pass $[\mathbf{FP}]$

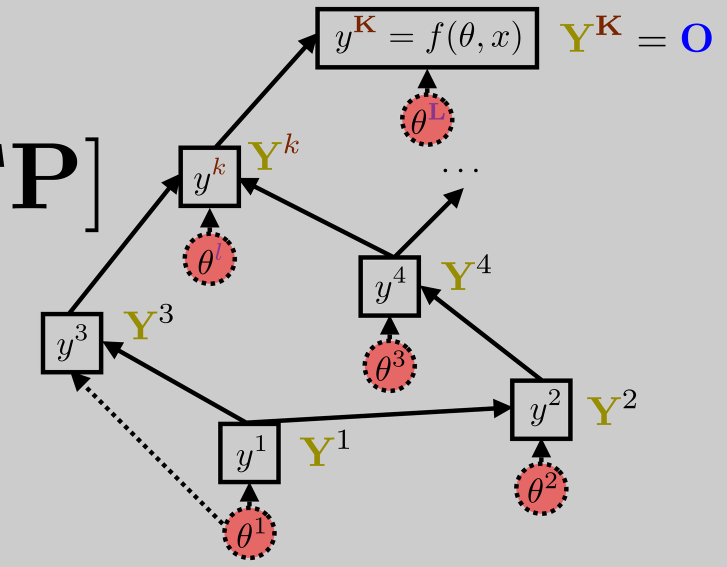


$$\Theta_\theta(x_1, x_2) = \frac{\partial f_1}{\partial \theta} \frac{\partial f_2}{\partial \theta}^T$$

Neural Network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$

Batch size $\mathbf{N}$ Output size $\mathbf{O}$

Forward Pass $[\mathbf{FP}]$



Parameters $\theta = \mathrm{vec}\left[\theta^0, \dots, \theta^{\mathbf{L}}\right]$, $\mathbf{P} = \sum_{l=0}^{\mathbf{L}} \mathbf{P}^l$

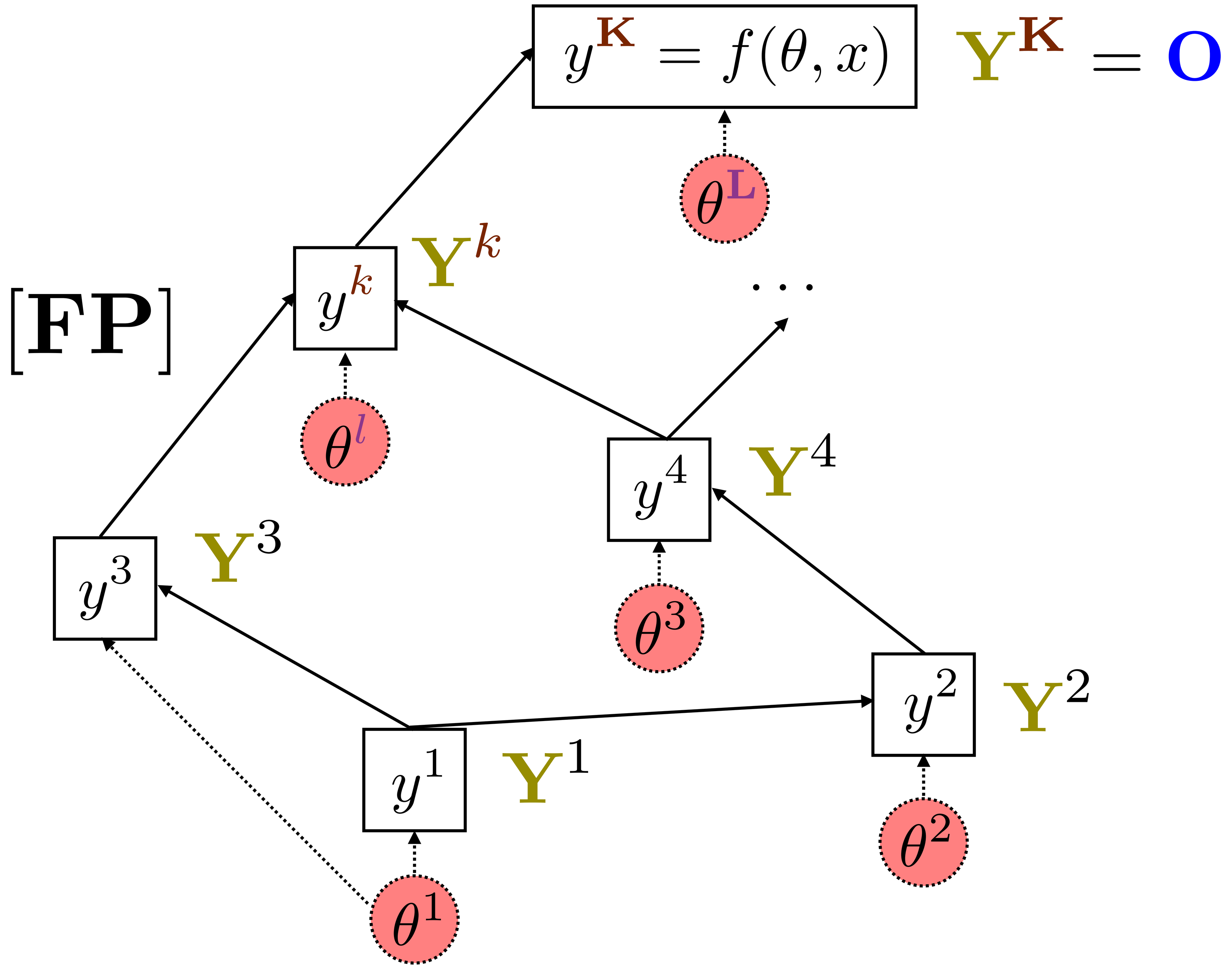
Intermediate primitive outputs $y^1, \dots, y^{\mathbf{K}}$, $\mathbf{Y} = \sum_{k=1}^{\mathbf{K}} \mathbf{Y}^k$

Stop the war
Stand with Ukraine

Idea 2: Structured derivatives

Apply the chain rule to the NTK expression:

Forward Pass $[\mathbf{FP}]$

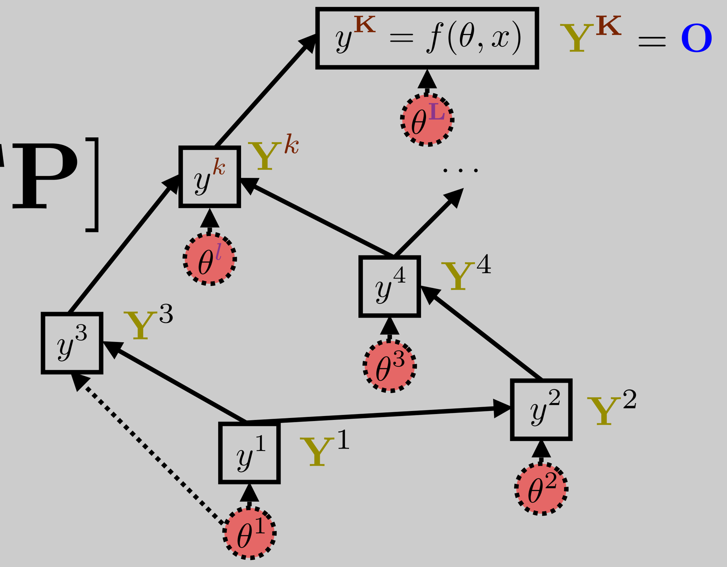


$$\Theta_\theta(x_1, x_2) = \frac{\partial f_1}{\partial \theta} \frac{\partial f_2}{\partial \theta}^T = \sum_{l=0}^{\mathbf{L}} \frac{\partial f_1}{\partial \theta^l} \frac{\partial f_2}{\partial \theta^l}^T =$$

Neural Network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$

Batch size $\mathbf{N}$ Output size $\mathbf{O}$

Forward Pass $[\mathbf{FP}]$



Parameters $\theta = \text{vec}\left[\theta^0, \dots, \theta^{\mathbf{L}}\right]$, $\mathbf{P} = \sum_{l=0}^{\mathbf{L}} \mathbf{P}^l$

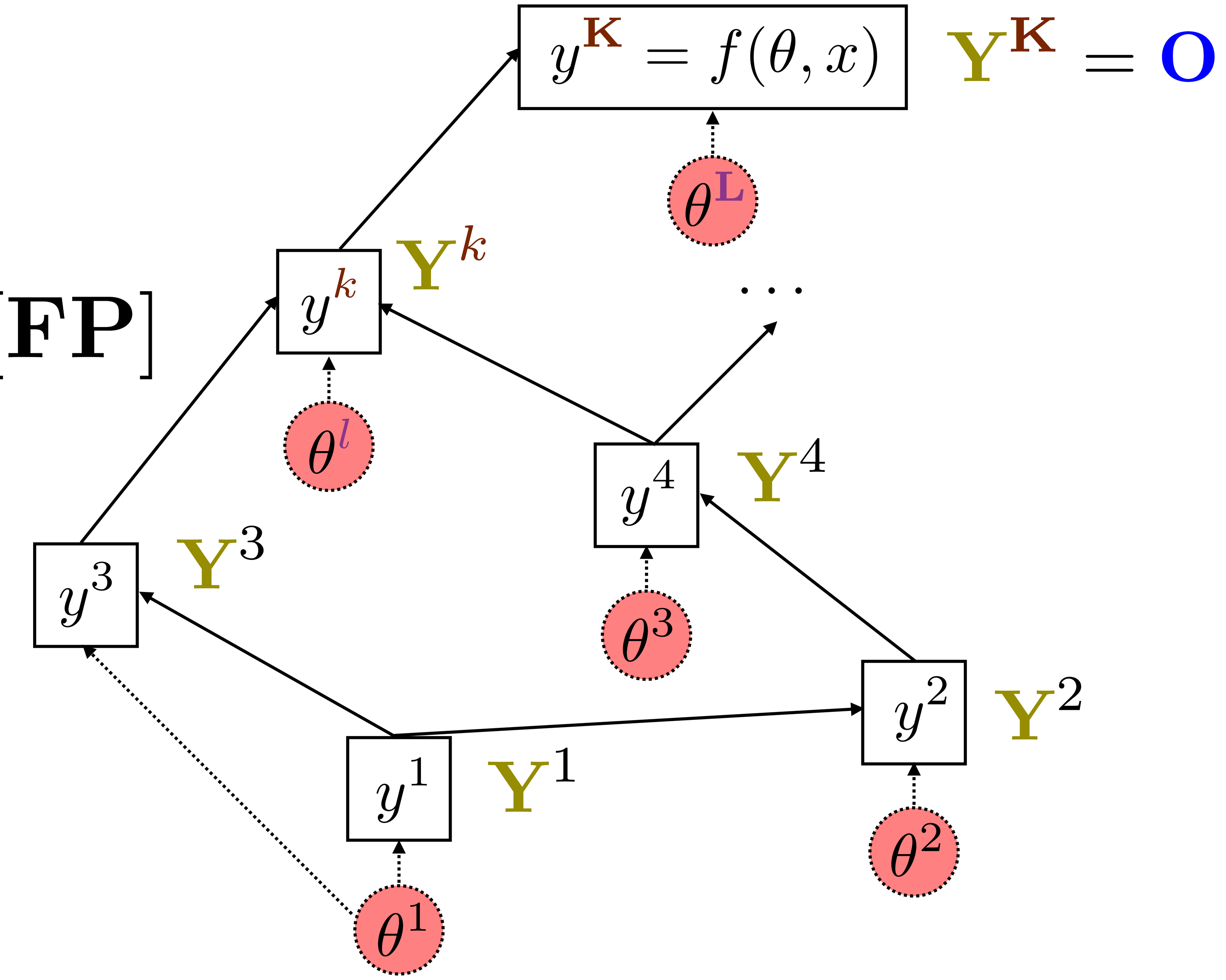
Intermediate primitive outputs $y^1, \dots, y^{\mathbf{K}}$, $\mathbf{Y} = \sum_{k=1}^{\mathbf{K}} \mathbf{Y}^k$

Stop the war
Stand with Ukraine

Idea 2: Structured derivatives

Apply the chain rule to the NTK expression:

Forward Pass $[\mathbf{FP}]$



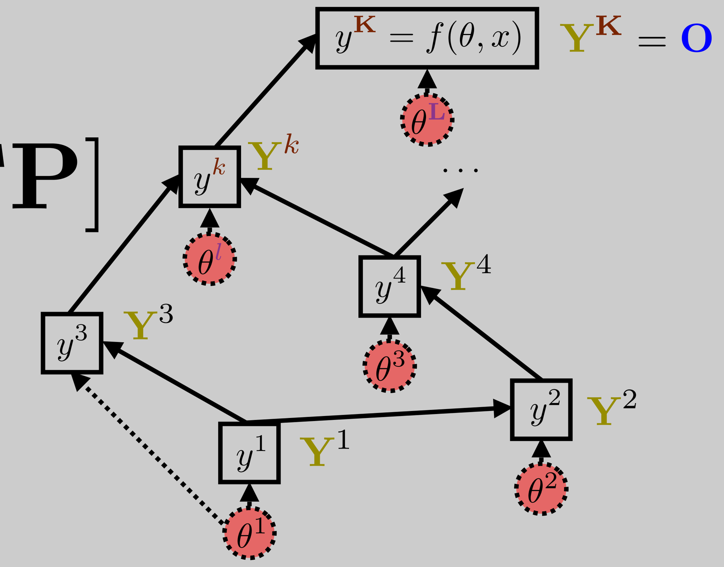
$$\Theta_\theta(x_1, x_2) = \frac{\partial f_1}{\partial \theta} \frac{\partial f_2}{\partial \theta}^T = \sum_{l=0}^{\mathbf{L}} \frac{\partial f_1}{\partial \theta^l} \frac{\partial f_2}{\partial \theta^l}^T =$$

$$= \sum_{l, k_1, k_2} \left(\frac{\partial f_1}{\partial y_1^{k_1}} \frac{\partial y_1^{k_1}}{\partial \theta^l} \right) \left(\frac{\partial f_2}{\partial y_2^{k_2}} \frac{\partial y_2^{k_2}}{\partial \theta^l} \right)^T =$$

Neural Network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$

Batch size $\mathbf{N}$ Output size $\mathbf{O}$

Forward Pass $[\mathbf{FP}]$



Parameters $\theta = \mathrm{vec}\left[\theta^0, \ldots, \theta^{\mathbf{L}}\right]$, $\mathbf{P} = \sum_{l=0}^{\mathbf{L}} \mathbf{P}^l$

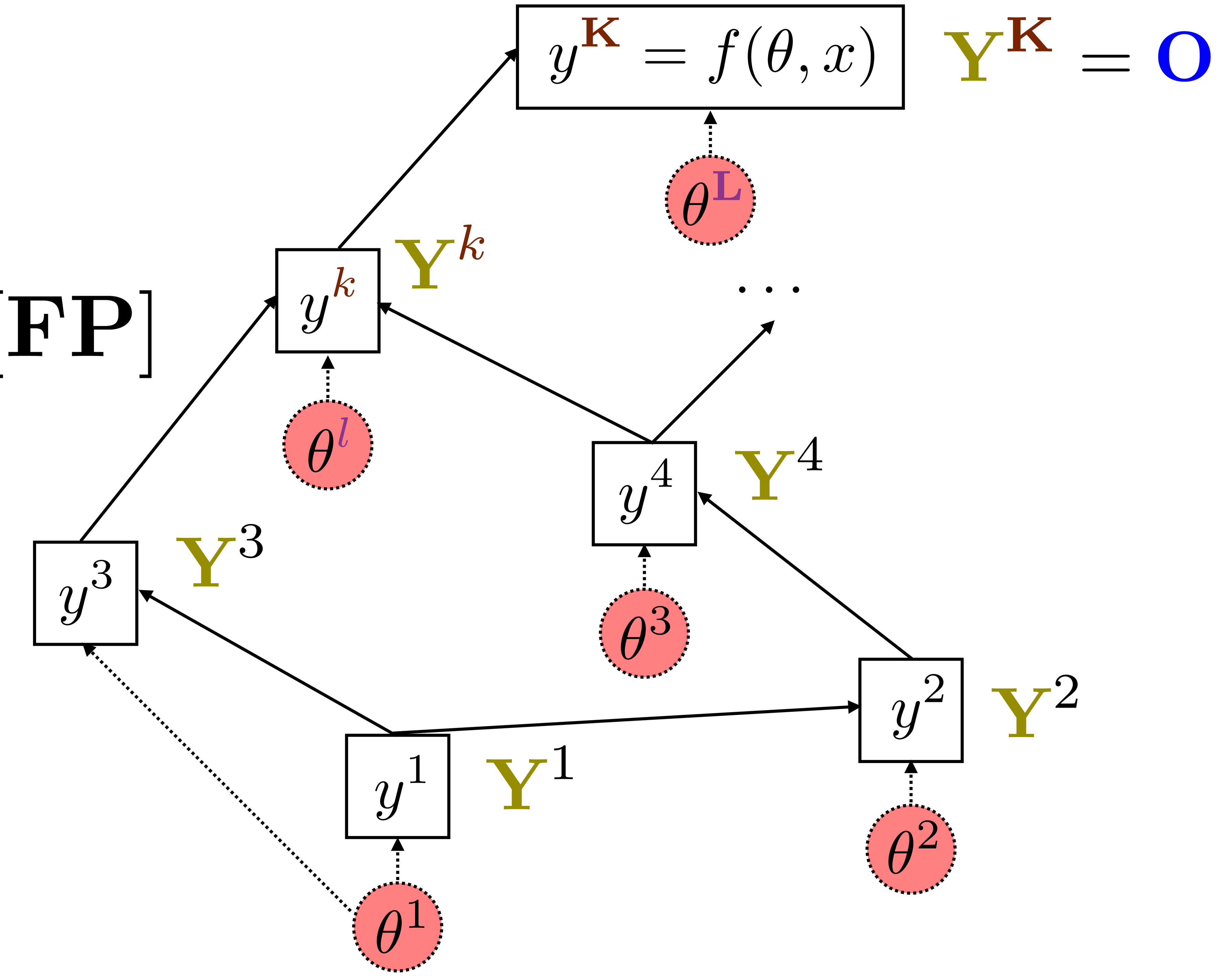
Intermediate primitive outputs $y^1, \ldots, y^{\mathbf{K}}$, $\mathbf{Y} = \sum_{k=1}^{\mathbf{K}} \mathbf{Y}^k$

Stop the war
Stand with Ukraine

Idea 2: Structured derivatives

Apply the chain rule to the NTK expression:

Forward Pass $[\mathbf{FP}]$



$$\Theta_\theta(x_1, x_2) = \frac{\partial f_1}{\partial \theta}\frac{\partial f_2}{\partial \theta}^T = \sum_{l=0}^{\mathbf{L}} \frac{\partial f_1}{\partial \theta^l}\frac{\partial f_2}{\partial \theta^l}^T =$$

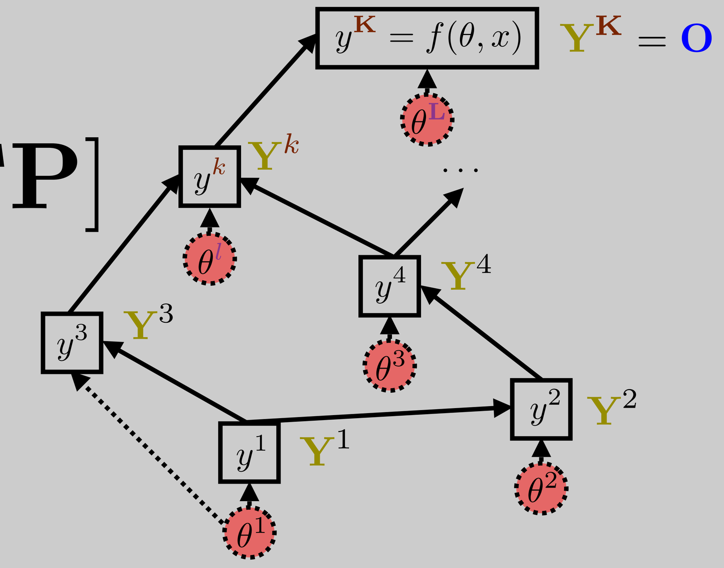
$$= \sum_{l, k_1, k_2} \left(\frac{\partial f_1}{\partial y_1^{k_1}}\frac{\partial y_1^{k_1}}{\partial \theta^l}\right)\left(\frac{\partial f_2}{\partial y_2^{k_2}}\frac{\partial y_2^{k_2}}{\partial \theta^l}\right)^T =$$

$$= \sum_{l, k_1, k_2} \frac{\partial f_1}{\partial y_1^{k_1}}\frac{\partial y_1^{k_1}}{\partial \theta^l}\frac{\partial y_2^{k_2}}{\partial \theta^l}^T \frac{\partial f_2}{\partial y_2^{k_2}}^T$$

Neural Network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$

Batch size $\mathbf{N}$ Output size $\mathbf{O}$

Forward Pass $[\mathbf{FP}]$



Parameters $\theta = \mathrm{vec}\left[\theta^0, \ldots, \theta^{\mathbf{L}}\right]$, $\mathbf{P} = \sum_{l=0}^{\mathbf{L}} \mathbf{P}^l$

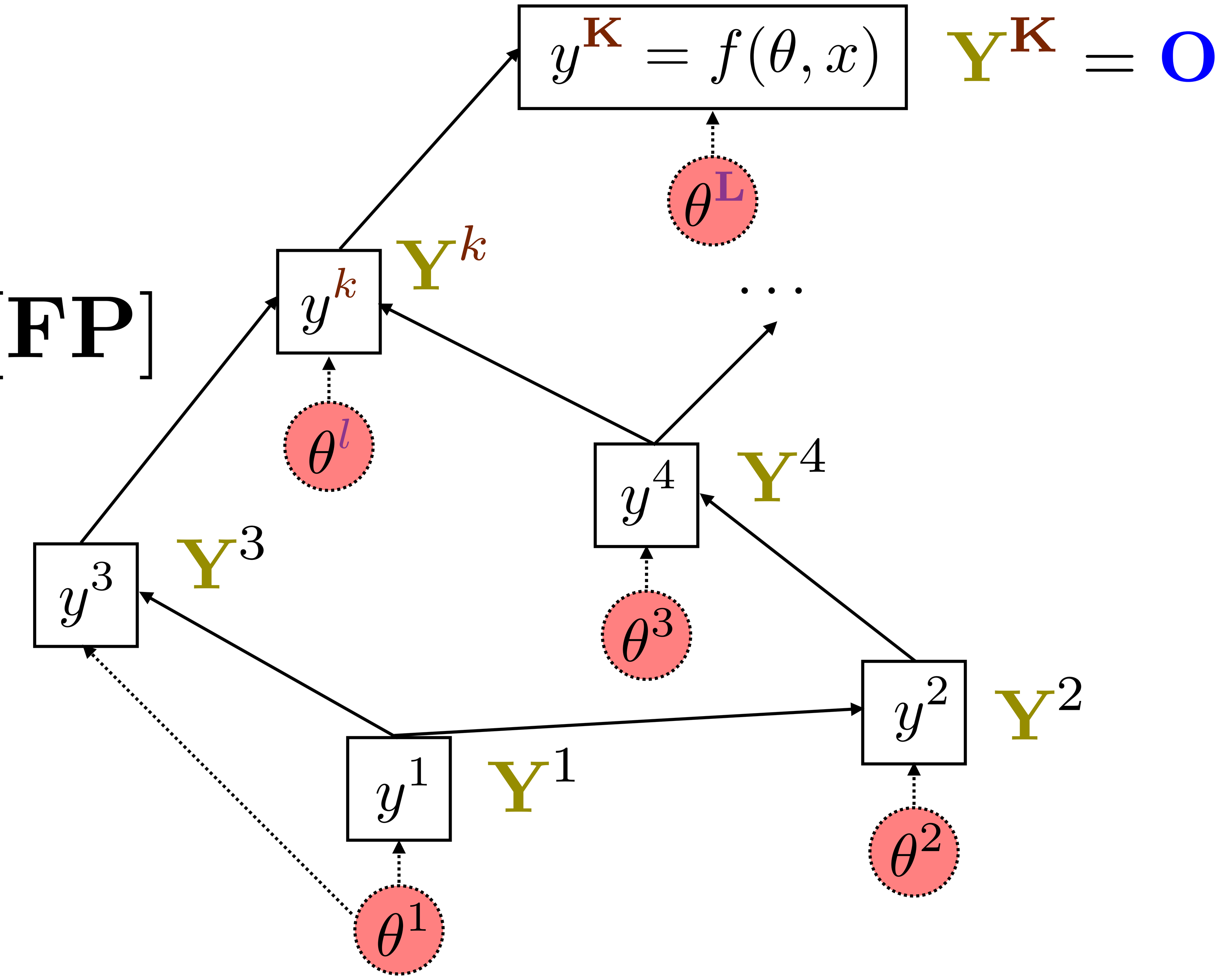
Intermediate primitive outputs $y^1, \ldots, y^{\mathbf{K}}$, $\mathbf{Y} = \sum_{k=1}^{\mathbf{K}} \mathbf{Y}^k$

Stop the war
Stand with Ukraine

Idea 2: Structured derivatives

Apply the chain rule to the NTK expression:

Forward Pass $[\mathbf{FP}]$



$$\Theta_\theta(x_1, x_2) = \frac{\partial f_1}{\partial \theta} \frac{\partial f_2}{\partial \theta}^T = \sum_{l=0}^{\mathbf{L}} \frac{\partial f_1}{\partial \theta^l} \frac{\partial f_2}{\partial \theta^l}^T =$$

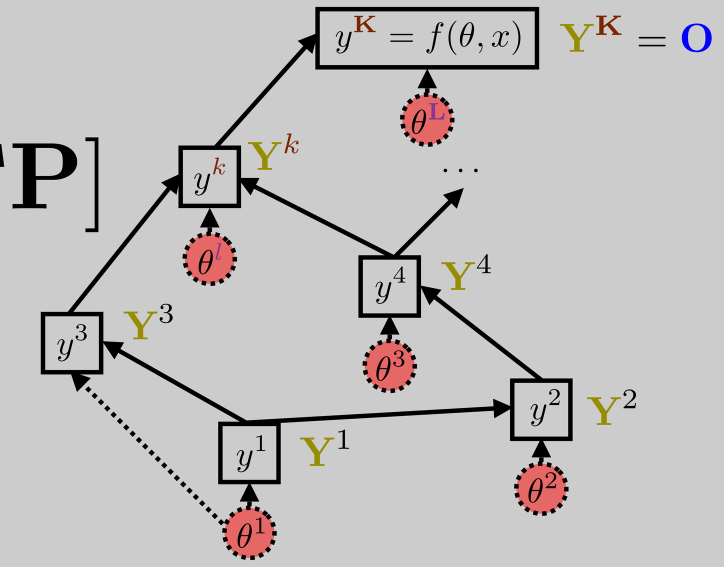
$$= \sum_{l,k_1,k_2} \left(\frac{\partial f_1}{\partial y_1^{k_1}} \frac{\partial y_1^{k_1}}{\partial \theta^l} \right) \left(\frac{\partial f_2}{\partial y_2^{k_2}} \frac{\partial y_2^{k_2}}{\partial \theta^l} \right)^T =$$

$$= \sum_{l,k_1,k_2} \frac{\partial f_1}{\partial y_1^{k_1}} \frac{\partial y_1^{k_1}}{\partial \theta^l} \frac{\partial y_2^{k_2}}{\partial \theta^l}^T \frac{\partial f_2}{\partial y_2^{k_2}}^T$$

Neural Network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$

Batch size $\mathbf{N}$ Output size $\mathbf{O}$

Forward Pass $[\mathbf{FP}]$



Parameters $\theta = \mathrm{vec}\left[\theta^0, \dots, \theta^{\mathbf{L}}\right],$ $\mathbf{P} = \sum_{l=0}^{\mathbf{L}} \mathbf{P}^l$

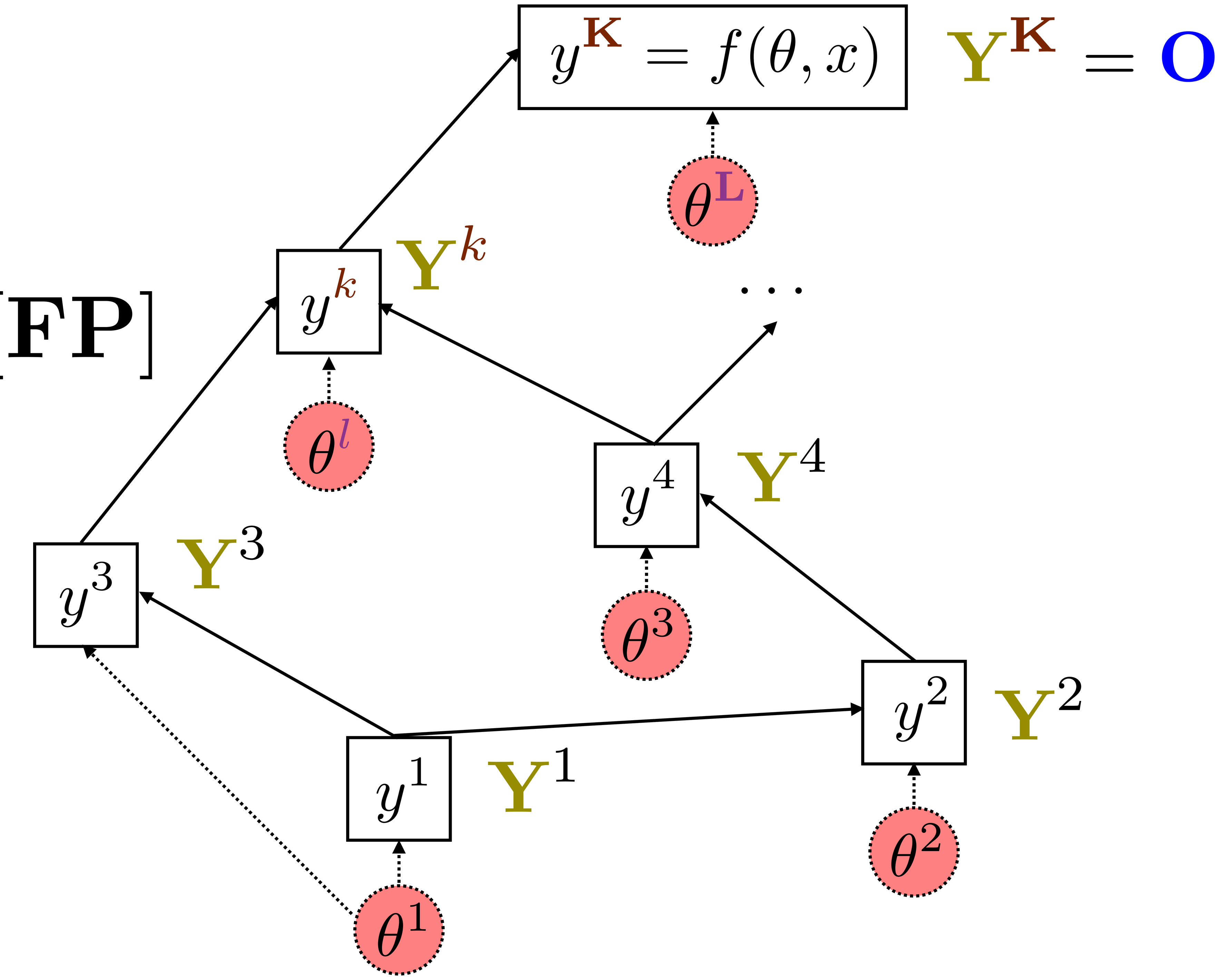
Intermediate primitive outputs $y^1, \dots, y^{\mathbf{K}},$ $\mathbf{Y} = \sum_{k=1}^{\mathbf{K}} \mathbf{Y}^k$

Stop the war
Stand with Ukraine

Idea 2: Structured derivatives

NTK is a sum of matrix-Jacobian-Jacobian-matrix products.

Forward Pass $[\mathbf{FP}]$



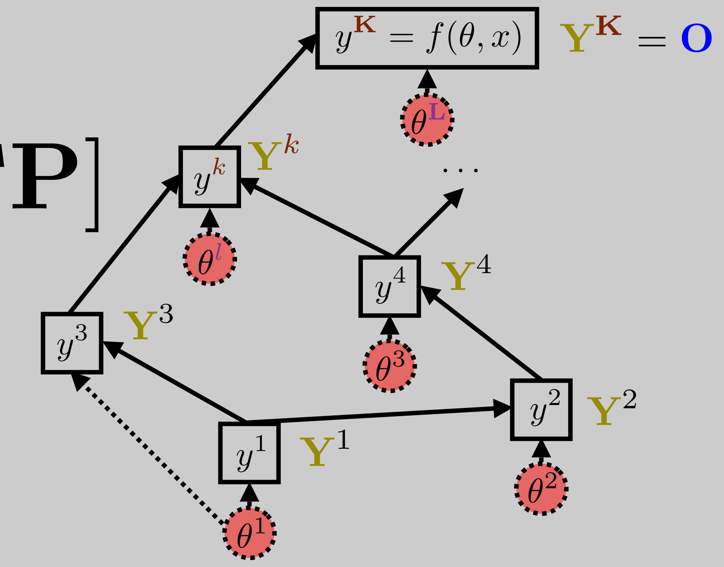
$$\Theta_\theta(x_1, x_2) = \frac{\partial f_1}{\partial \theta}\frac{\partial f_2}{\partial \theta}^T = \sum_{l=0}^{\mathbf{L}} \frac{\partial f_1}{\partial \theta^l}\frac{\partial f_2}{\partial \theta^l}^T = \sum_{l,k_1,k_2}\left(\frac{\partial f_1}{\partial y_1^{k_1}}\frac{\partial y_1^{k_1}}{\partial \theta^l}\right)\left(\frac{\partial f_2}{\partial y_2^{k_2}}\frac{\partial y_2^{k_2}}{\partial \theta^l}\right)^T$$

"matrix-Jacobian-Jacobian-matrix product" (MJJMP)

$$= \sum_{l,k_1,k_2} \frac{\partial f_1}{\partial y_1^{k_1}}\frac{\partial y_1^{k_1}}{\partial \theta^l}\frac{\partial y_2^{k_2}}{\partial \theta^l}^T\frac{\partial f_2}{\partial y_2^{k_2}}^T$$

Neural Network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$

Forward Pass $[\mathbf{FP}]$



Parameters $\theta = \mathrm{vec}\left[\theta^0, \ldots, \theta^{\mathbf{L}}\right]$, $\mathbf{P} = \sum_{l=0}^{\mathbf{L}} \mathbf{P}^l$

Batch size $\mathbf{N}$ Output size $\mathbf{O}$

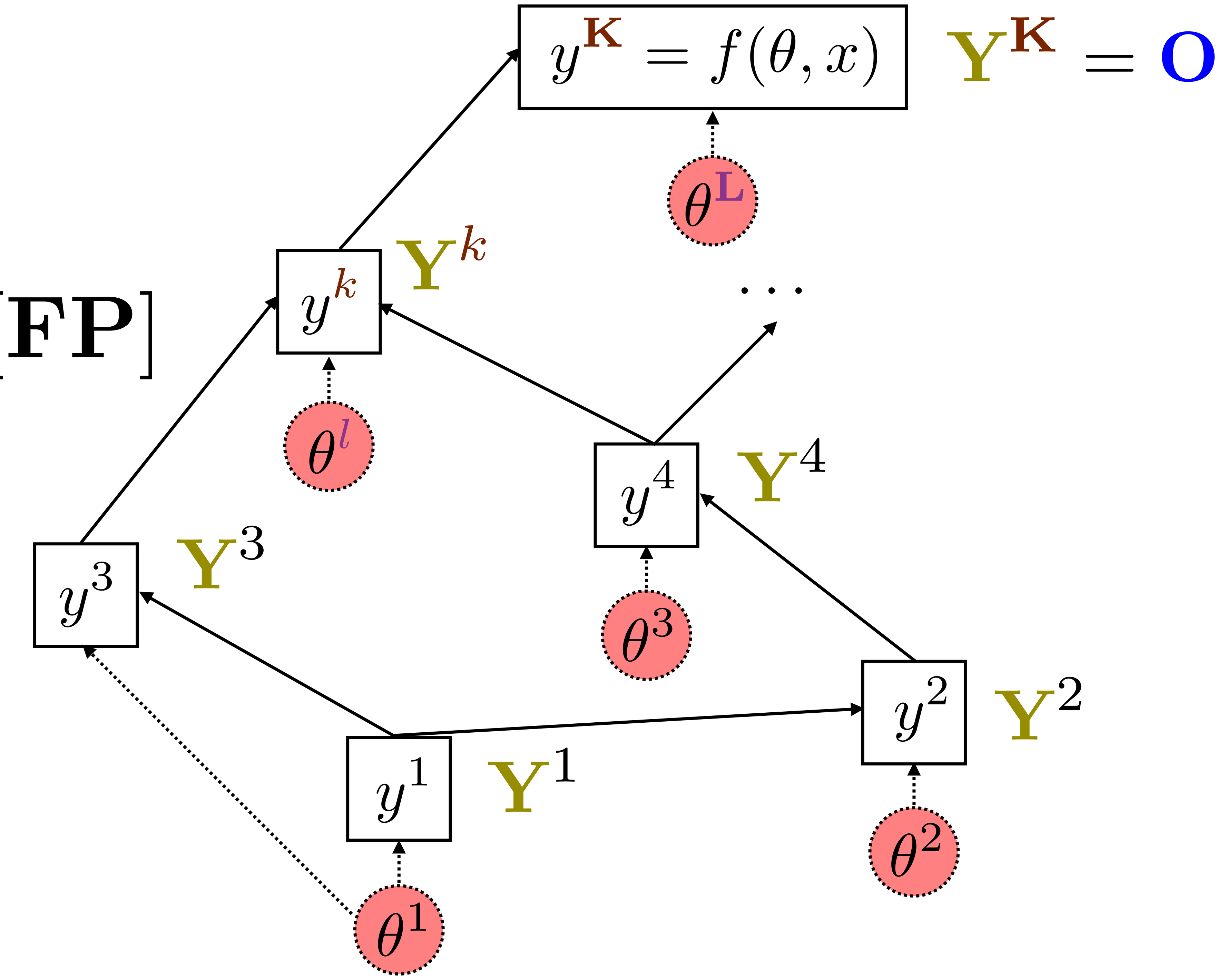
Intermediate primitive outputs $y^1, \ldots, y^{\mathbf{K}}$, $\mathbf{Y} = \sum_{k=1}^{\mathbf{K}} \mathbf{Y}^k$

Stop the war
Stand with Ukraine

Idea 2: Structured derivatives

Idea: for every pair of primitives y_1, y_2, implement efficient MJJMPs:

Forward Pass $[\mathbf{FP}]$



$$\Big(\underbrace{c_1}_{\mathbf{O}\times\mathbf{Y}_1}, \underbrace{c_2}_{\mathbf{O}\times\mathbf{Y}_2}\Big) \mapsto \underbrace{c_1 \frac{\partial y_1}{\partial \theta} \frac{\partial y_2}{\partial \theta}^T c_2^T}_{\mathbf{O}\times\mathbf{O}}$$

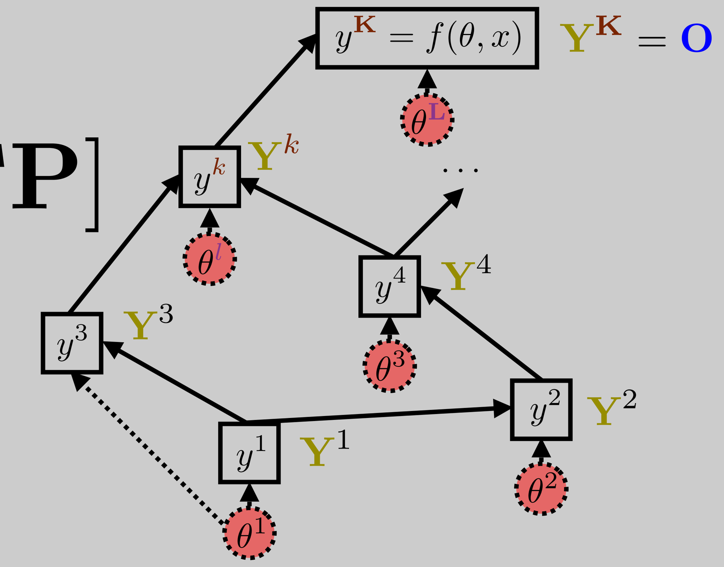
"matrix-Jacobian-Jacobian-matrix product" (MJJMP)

$$= \sum_{l,k_1,k_2} \frac{\partial f_1}{\partial y_1^{k_1}} \frac{\partial y_1^{k_1}}{\partial \theta^l} \frac{\partial y_2^{k_2}}{\partial \theta^l}^T \frac{\partial f_2}{\partial y_2^{k_2}}^T$$

Neural Network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$

Batch size $\mathbf{N}$ Output size $\mathbf{O}$

Forward Pass $[\mathbf{FP}]$



Parameters $\theta = \mathrm{vec}\left[\theta^0, \ldots, \theta^{\mathbf{L}}\right]$, $\mathbf{P} = \sum_{l=0}^{\mathbf{L}} \mathbf{P}^l$

Intermediate primitive outputs $y^1, \ldots, y^{\mathbf{K}}$, $\mathbf{Y} = \sum_{k=1}^{\mathbf{K}} \mathbf{Y}^k$

Stop the war
Stand with Ukraine

Idea 2: Structured derivatives

Idea: for every pair of primitives y_1, y_2, implement efficient **MJJMPs**:

$$\Big(\underbrace{c_1}_{\mathbf{O}\times\mathbf{Y}_1}, \underbrace{c_2}_{\mathbf{O}\times\mathbf{Y}_2}\Big) \mapsto \underbrace{c_1 \frac{\partial y_1}{\partial \theta} \frac{\partial y_2}{\partial \theta}^T c_2^{\,T}}_{\mathbf{O}\times\mathbf{O}}$$

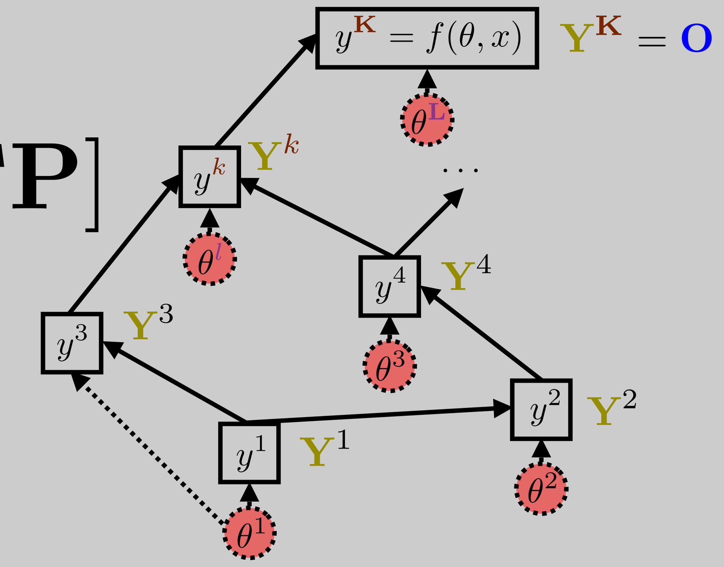
Similar to existing AD tools: for every primitive y, implement efficient

$$\mathrm{JVP}_{(y,\theta)} : \theta_t \in \mathbb{R}^{\mathbf{P}} \mapsto \frac{\partial y}{\partial \theta}\theta_t \in \mathbb{R}^{\mathbf{Y}} \qquad \mathrm{VJP}_{(y,\theta)} : y_c \in \mathbb{R}^{\mathbf{Y}} \mapsto \frac{\partial y}{\partial \theta}^T y_c \in \mathbb{R}^{\mathbf{P}}$$

Neural Network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$

Batch size $\mathbf{N}$ Output size $\mathbf{O}$

Forward Pass $[\mathbf{FP}]$



Parameters $\theta = \mathrm{vec}\left[\theta^0, \ldots, \theta^{\mathbf{L}}\right]$, $\mathbf{P} = \sum_{l=0}^{\mathbf{L}} \mathbf{P}^l$

Intermediate primitive outputs $y^1, \ldots, y^{\mathbf{K}}$, $\mathbf{Y} = \sum_{k=1}^{\mathbf{K}} \mathbf{Y}^k$

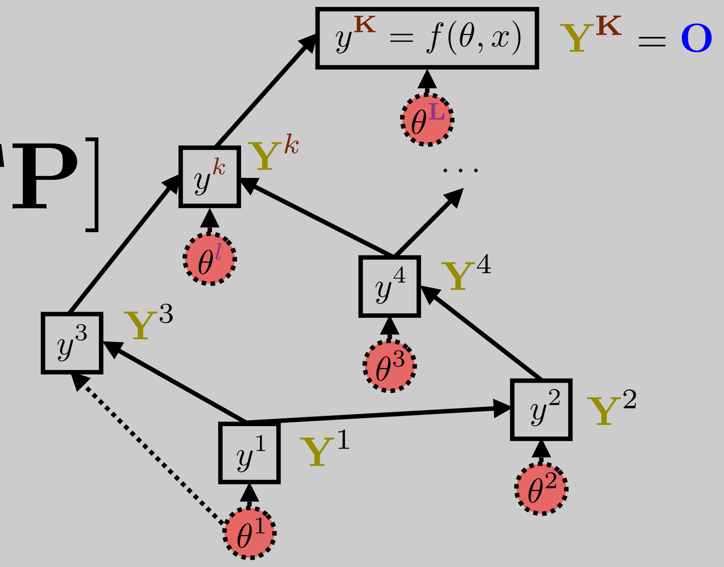
Stop the war
Stand with Ukraine

Methods summary

Method	Jacobian Contraction	NTK-vector products	Structured derivatives
Time	$\mathbf{N}\mathbf{O}\,[\mathbf{FP}] + \mathbf{N}^2\mathbf{O}^2\mathbf{P}$	$\mathbf{N}^2\mathbf{O}\,[\mathbf{FP}]$	$\mathbf{N}\mathbf{O}\,[\mathbf{FP}] + \mathbf{MJJMP}$

Neural Network $f(\theta, x) \in \mathbb{R}^{\mathbf{O}}$

Forward Pass $[\mathbf{FP}]$



Parameters $\theta = \mathrm{vec}\left[\theta^0, \ldots, \theta^{\mathbf{L}}\right]$, $\mathbf{P} = \sum_{l=0}^{\mathbf{L}} \mathbf{P}^l$

Batch size $\mathbf{N}$ Output size $\mathbf{O}$

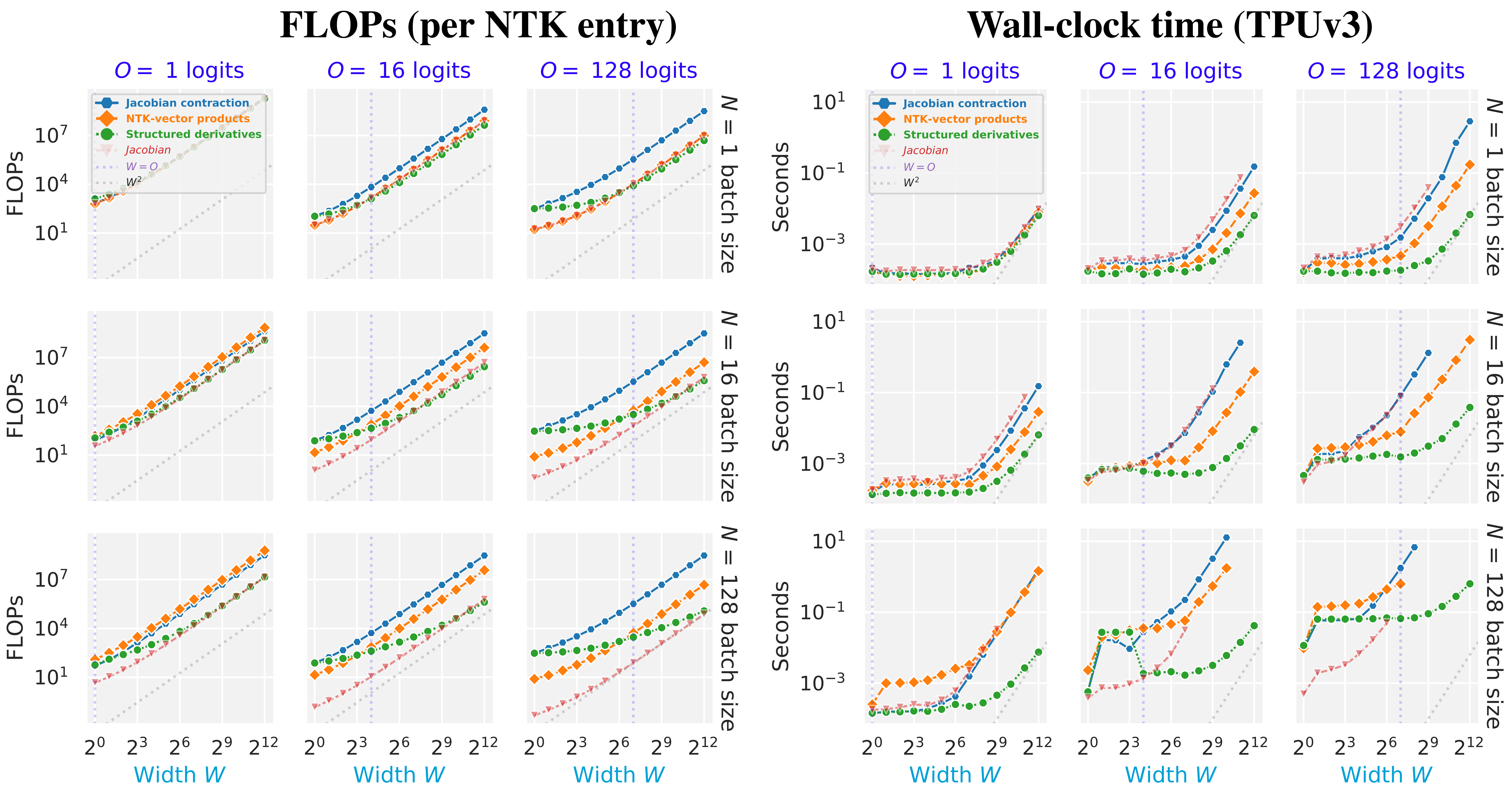
Intermediate primitive outputs $y^1, \ldots, y^{\mathbf{K}}$, $\mathbf{Y} = \sum_{k=1}^{\mathbf{K}} \mathbf{Y}^k$

Plan

1. NTK definition and notation
2. Recap of Automatic Differentiation (AD)
3. Baseline NTK computation complexity
4. Two new algorithms for computing the NTK
5. **Benchmarks**

Stop the war
Stand with Ukraine

Benchmarks on FCNs



Stop the war
Stand with Ukraine

Benchmarks on ImageNet ResNets

Stop the war
Stand with Ukraine

ImageNet Transformers/Mixers/Hybrids

Stop the war
Stand with Ukraine

Stop the war
Stand with Ukraine

2-line code sample

```
pip install neural_tangents
```

```
import neural_tangents as nt


# Given any f: params, x -> f(params, x)
ntk_fn = nt.empirical_ntk_fn(f, implementation=1)  # 1, 2, or 3
ntk = ntk_fn(x1, x2, params)
```

github.com/google/neural-tangents


Stop the war
Stand with Ukraine
SupportUkraineNow.org