A DALL·E 3 artwork

Transformers with built-in IRC safety in particle physics

Congqiao Li *(Peking University)*

Honouring the input from
Huilin Qu[2], Sitian Qian[1], Leyun Gao[1], Qiang Li[1]
[1]PKU [2]CERN

ML4Jets 2023 · Hamburg
8 November, 2023

Background and introduction

IRC safety

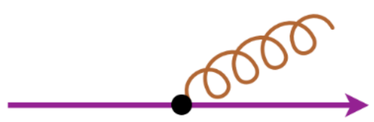
- ➔ Jet observables are preferred to be IRC safe as it is tractable in pQCD theory
- ➔ This motivates the design of an IRC-safe jet NN
  - ❖ its output scores will be IRC-safe observables
- ➔ Usually not emphasised in current experimental NN usage
  - ❖ wanting to achieve top experimental performance vs good theory interpretability → this is an experimental–theoretical dilemma
- ➔ NN for IRC safety: we take the practical definition:
  - ❖ i.e., NN output does not change when there are infinitesimal soft emissions or an exact collinear splitting

$$\lim_{\epsilon \to 0} f^{(N+1)}\left(\{p_1, \cdots, p_N, \epsilon p_{N+1}\}\right) = f^{(N)}\left(\{p_1, \cdots, p_N\}\right)$$ infrared safety

$$f^{(N+1)}\left(\{p_1, \cdots, \lambda p_N, (1-\lambda) p_N\}\right) = f^{(N)}\left(\{p_1, \cdots, p_N\}\right) \quad (\lambda \in [0,1])$$ collinear safety

$f^{(N)}(\,\cdot\,) =$ the NN function, when applied to jets with N particles

From a theory perspective, divergence seen in the QCD splitting function:
(soft emission or collinear splitting)



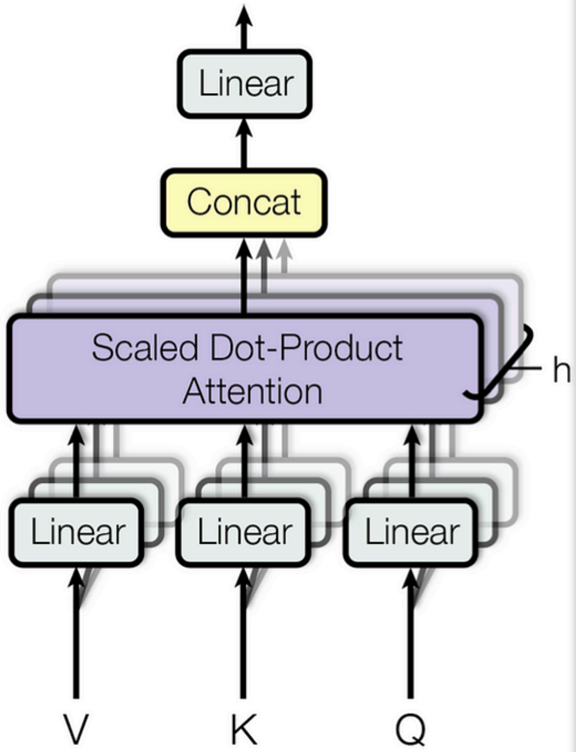
$$dP_{i\to ig} \simeq \frac{2\alpha_s}{\pi} C_i \frac{d\theta}{\theta} \frac{dz}{z}$$

[F. Tkachov, hep-ph/9601308]

- note: a theoretical definition of IRC safety is on C-correlators, and τ_{21} is not IRC safe (but "Sudakov safe")
  [A. Larkoski, S. Marzani, and J. Thaler, 1502.01719]
- in this work, we still stick to the practical definition like the other network designing works

Background and introduction

"A Transformer block"

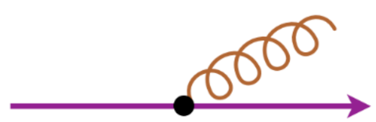


Transformers

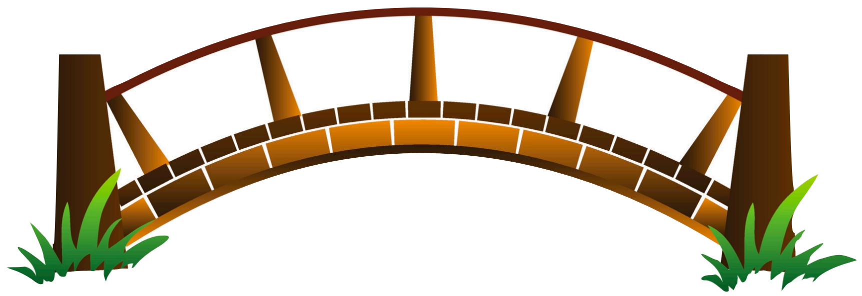
- → A general network architecture basically made up of multi-head attention blocks
- → It unifies the architecture designs in vision and language tasks, and was increasingly adopted in more fields!
- → Benefits:
  - ❖ efficiently learn relations of tokens
  - ❖ scale well on larger datasets
  - ❖ → often achieve SoTA performance
- → Relations to GNN?
  - ❖ like a fully connected graph, but the message passing achieved by a lightweight dot-product attention mechanism
- → Application in HEP
  - ❖ for jet tagging/regression etc.: using low-level particle inputs, where we just treat each particle as a token
  - ❖ in analysis-level: each object (jet/lepton) as a token

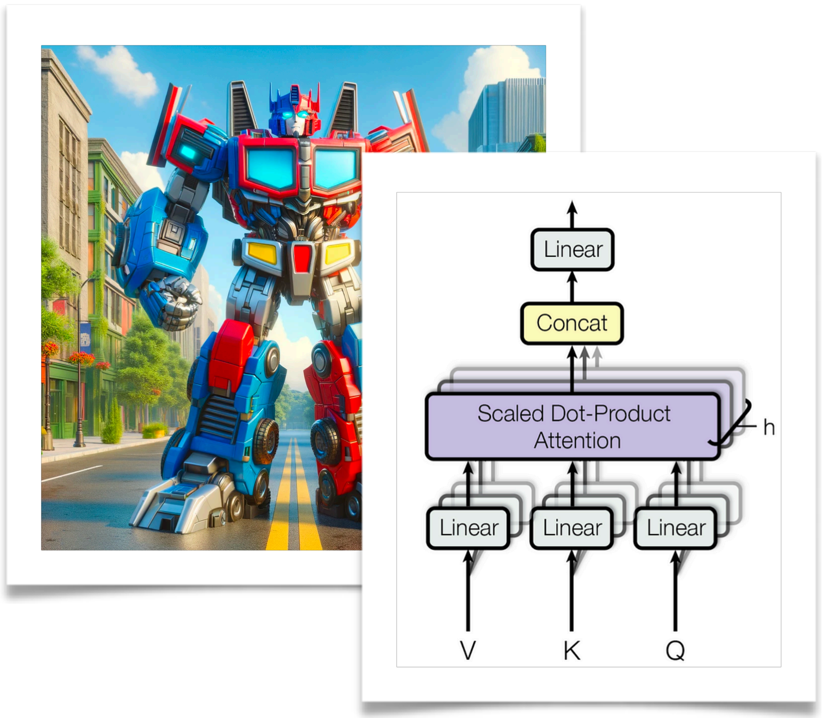
Bridging IRC safety with Transformer

Making an observable
(or NN output) IRC safe



$$dP_{i\to ig} \simeq \frac{2\alpha_s}{\pi} C_i \frac{d\theta}{\theta} \frac{dz}{z}$$



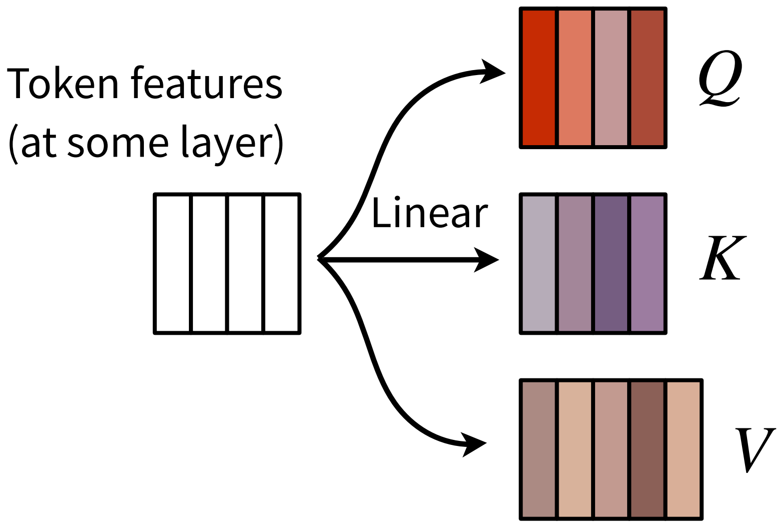


➔ In this talk, we will introduce a recipe to modify the general dot-product attention mechanism

- basically is a trick to build in IRC safety into particle-based Transformers
- the solution is a general one
  - e.g. also works with Transformers added with other ingredients (e.g. ParT, with additional pairwise features as attentive bias + class token)
- allow us to continue benefiting from the good performance brought by Transformers with a small performance trade-off
  - a possible choice for future experimental applications

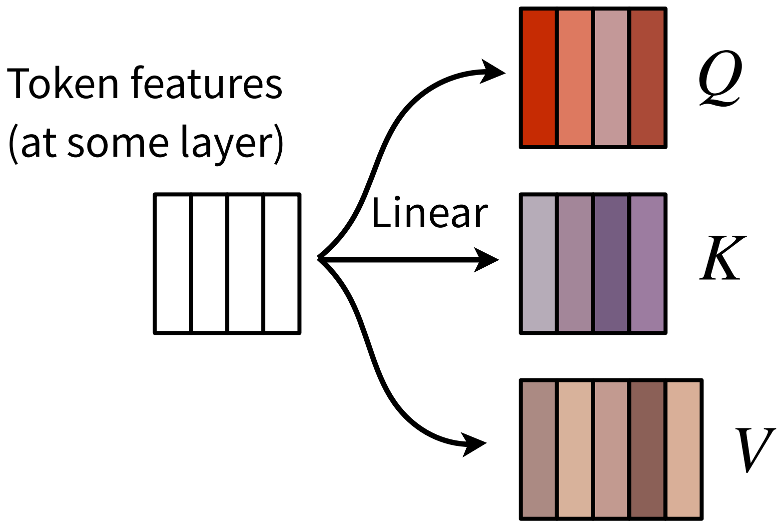
Revisiting dot-product attention

1. Derive Q, K, V

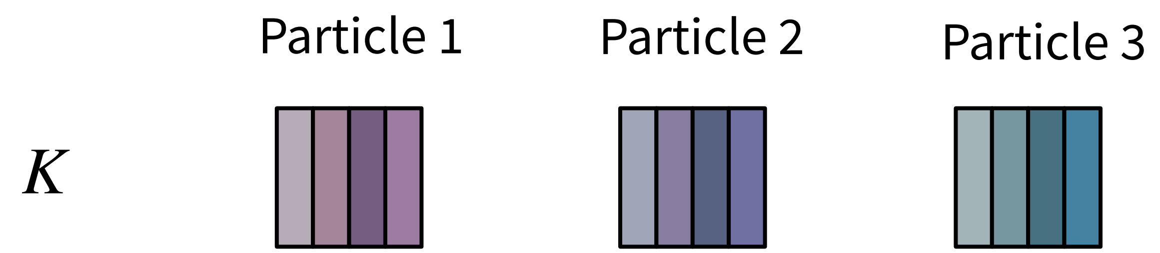


Revisiting dot-product attention

1. Derive Q, K, V

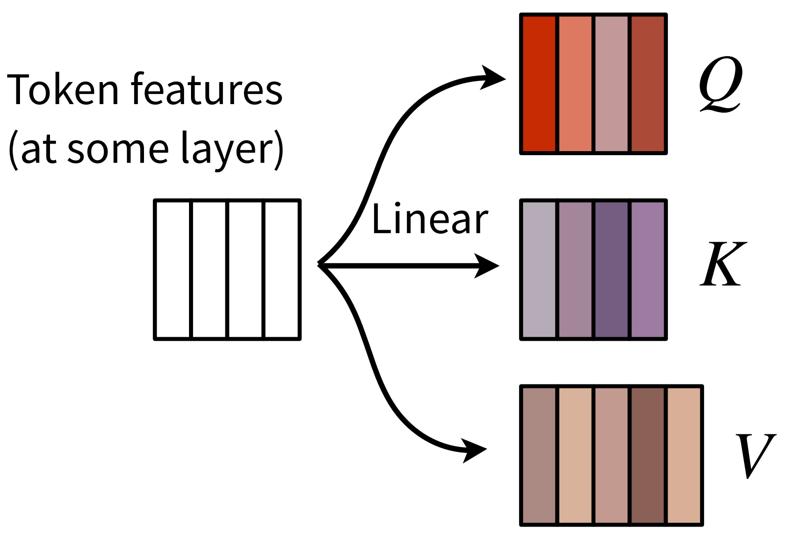


2. Calculate $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q^T K}{\sqrt{d}}\right)V$

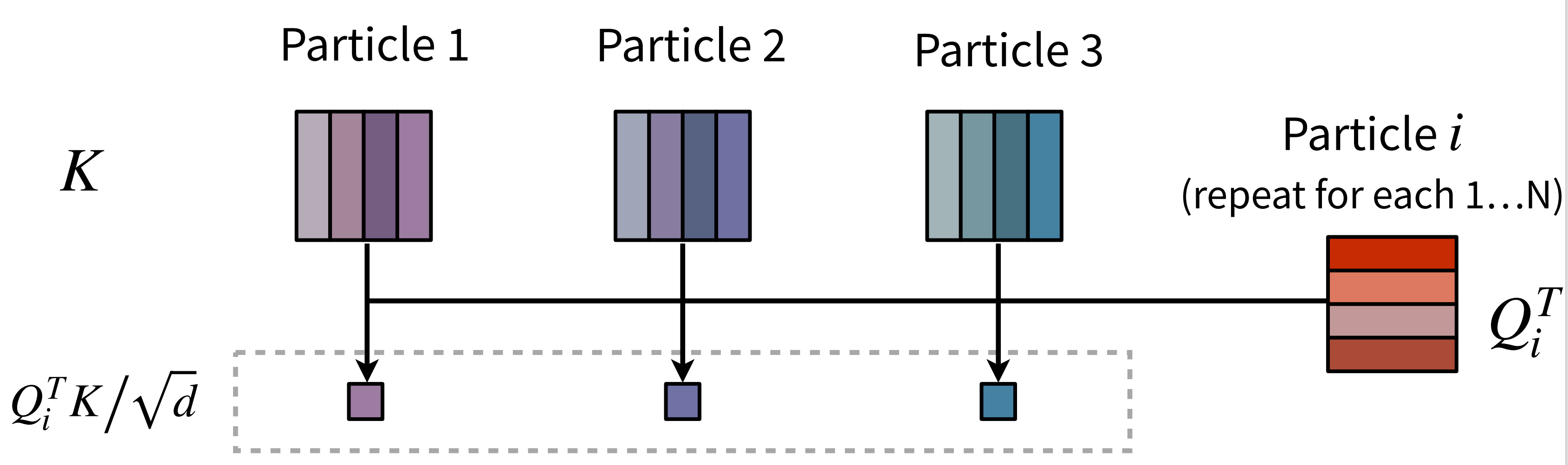


Revisiting dot-product attention

1. Derive Q, K, V

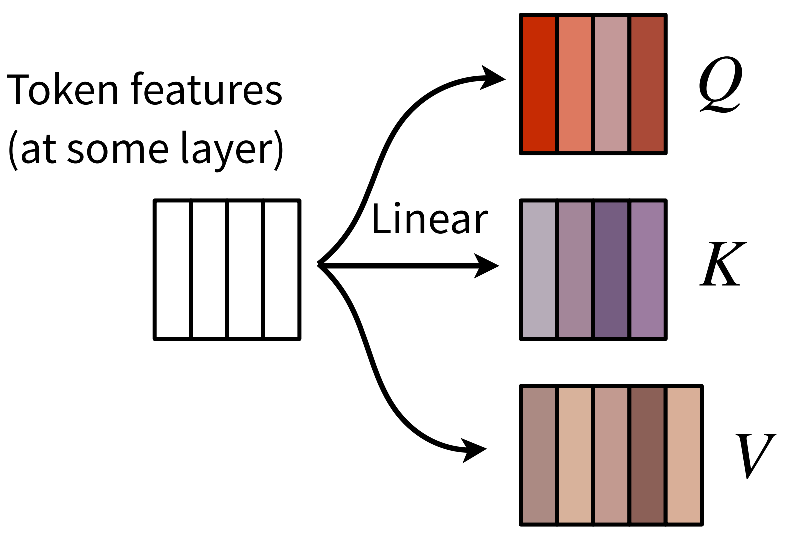


2. Calculate $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q^T K}{\sqrt{d}}\right) V$

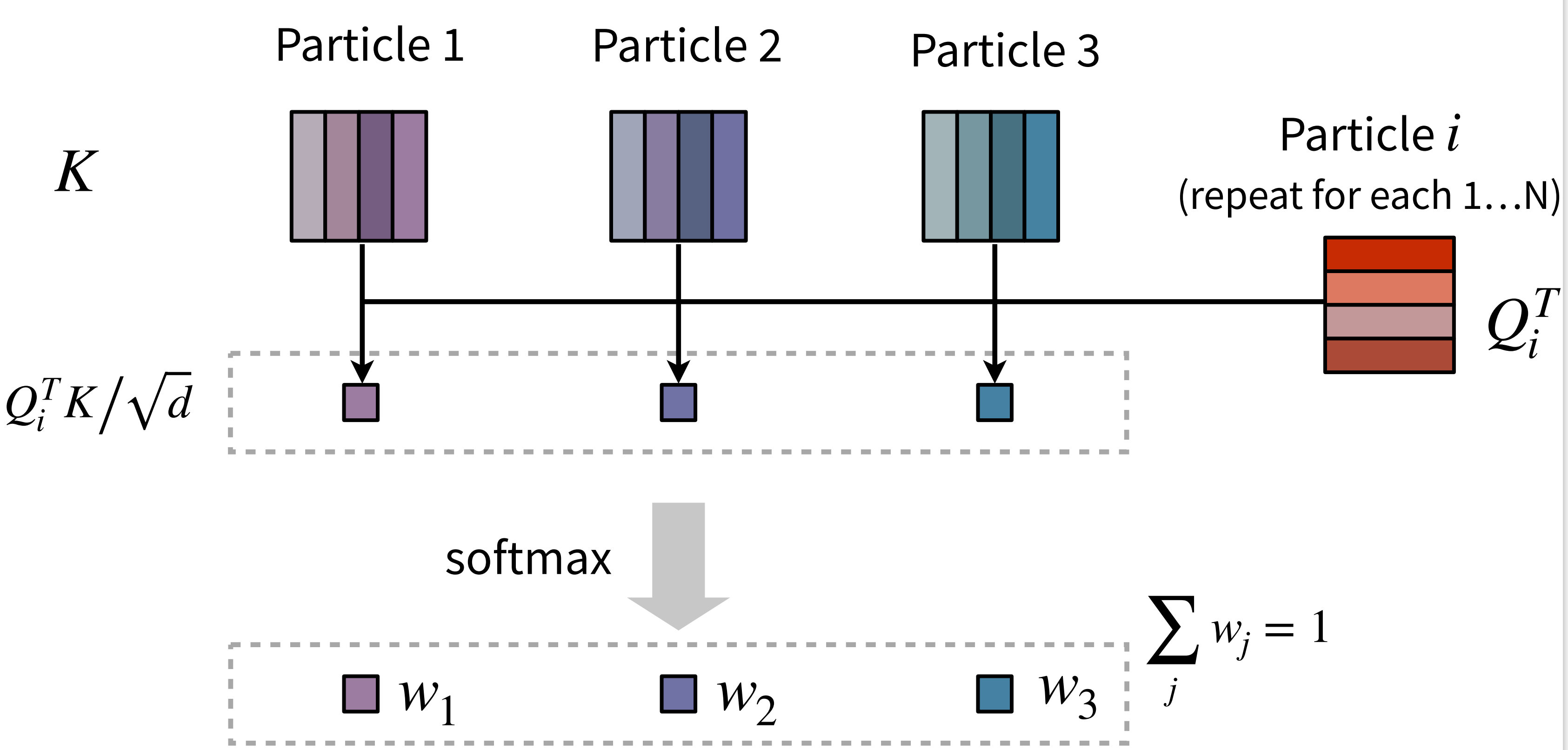


Revisiting dot-product attention

1. Derive Q, K, V

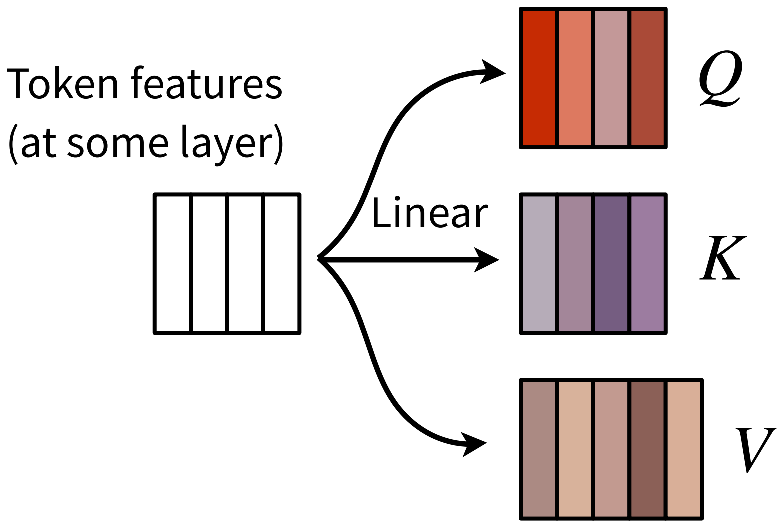


2. Calculate $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q^T K}{\sqrt{d}}\right) V$

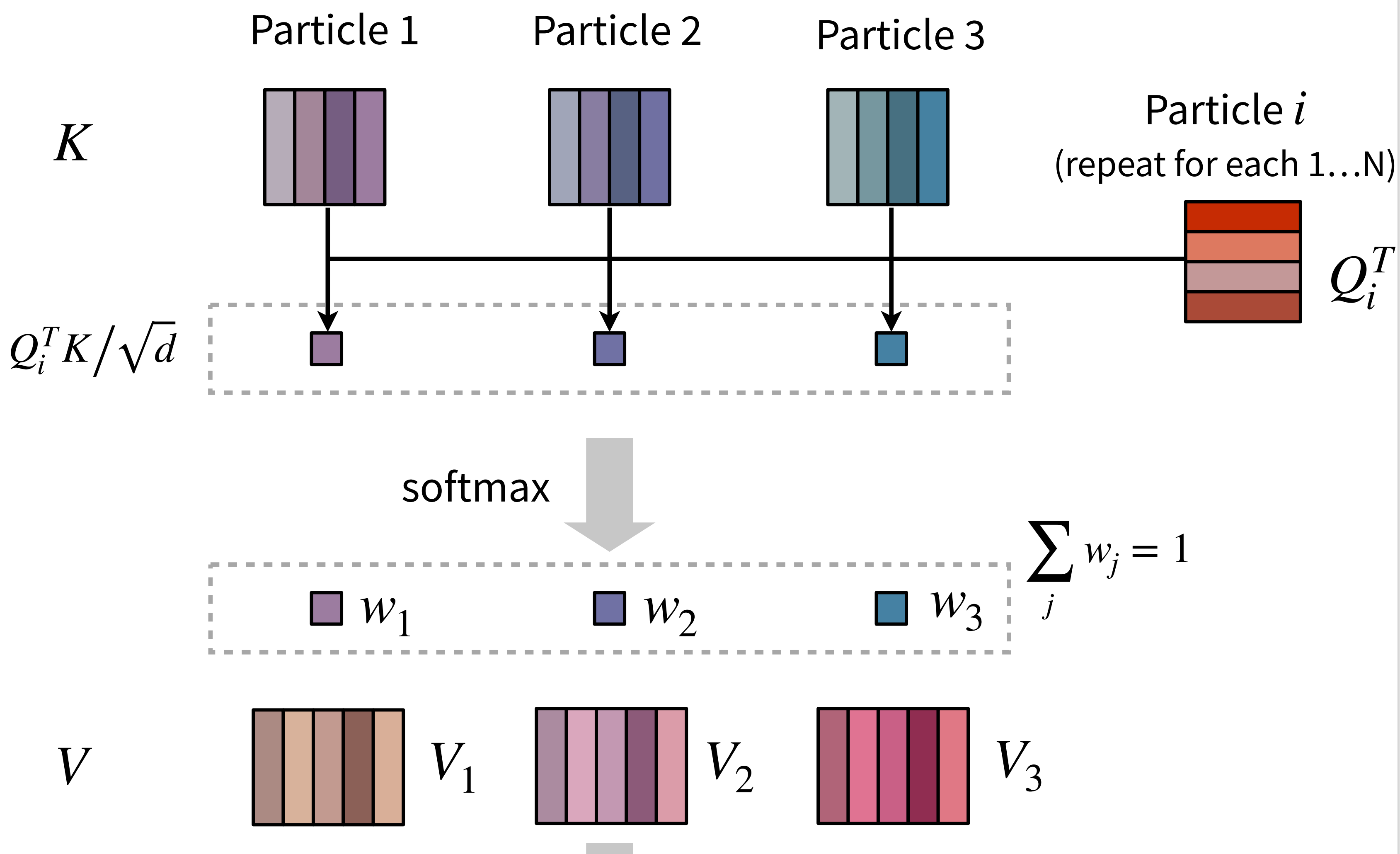


Revisiting dot-product attention

1. Derive Q, K, V



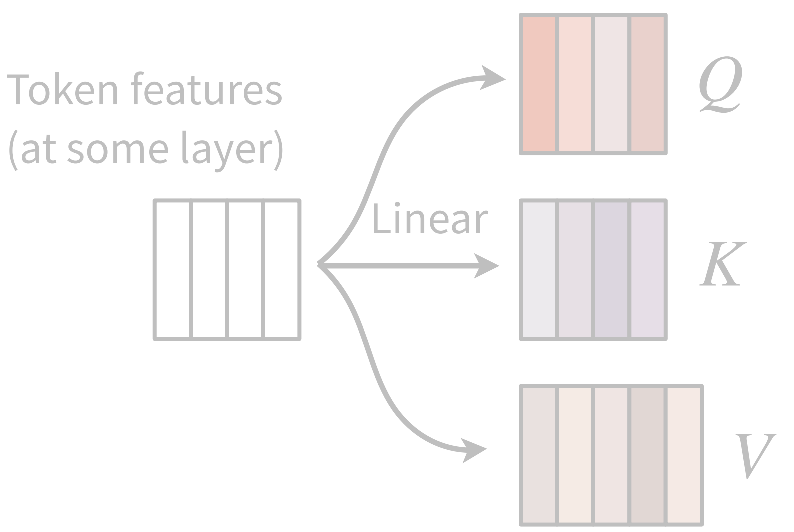
2. Calculate $\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{Q^T K}{\sqrt{d}}\right)V$



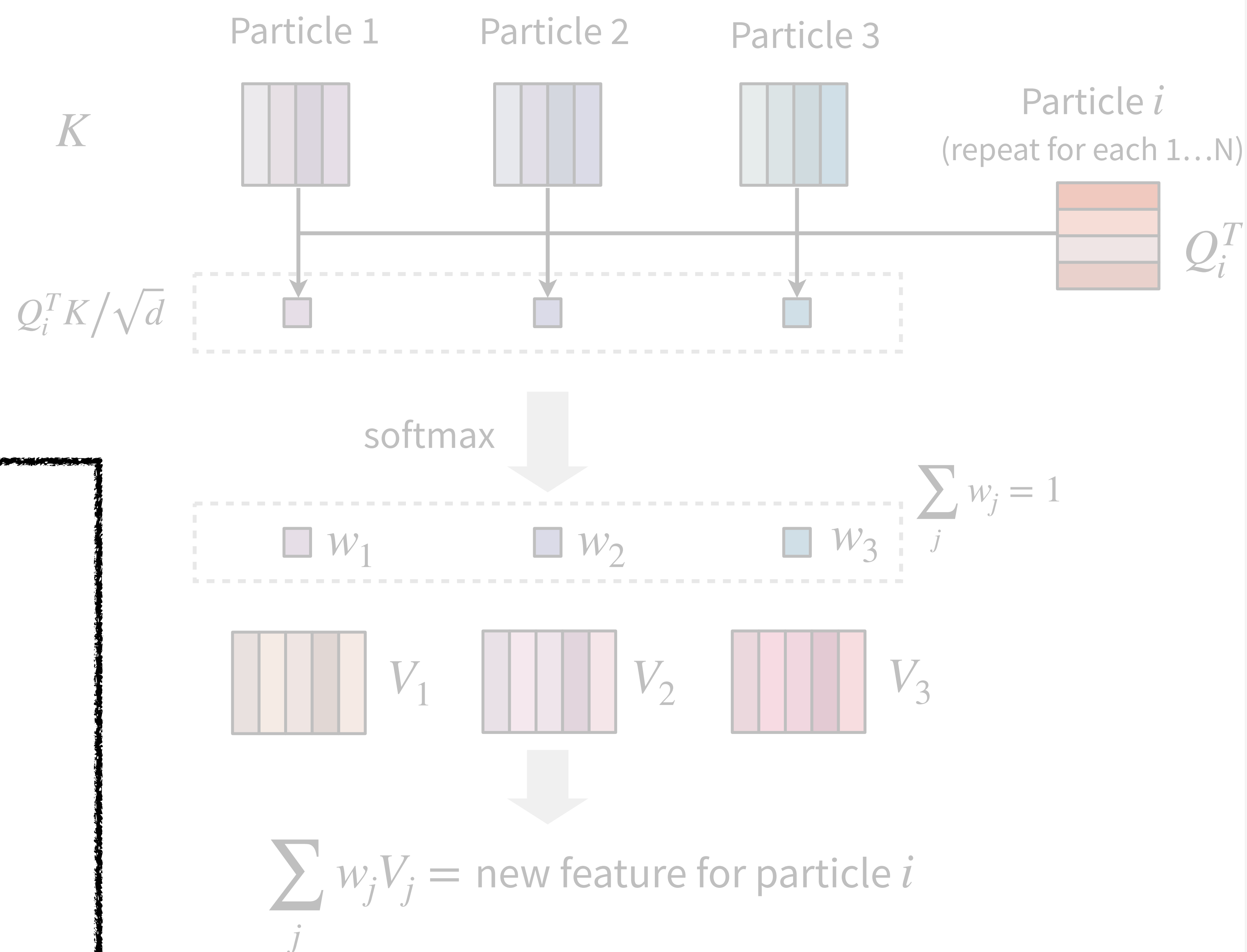
$\sum_j w_j V_j =$ new feature for particle i

Revisiting dot-product attention

1. Derive Q, K, V



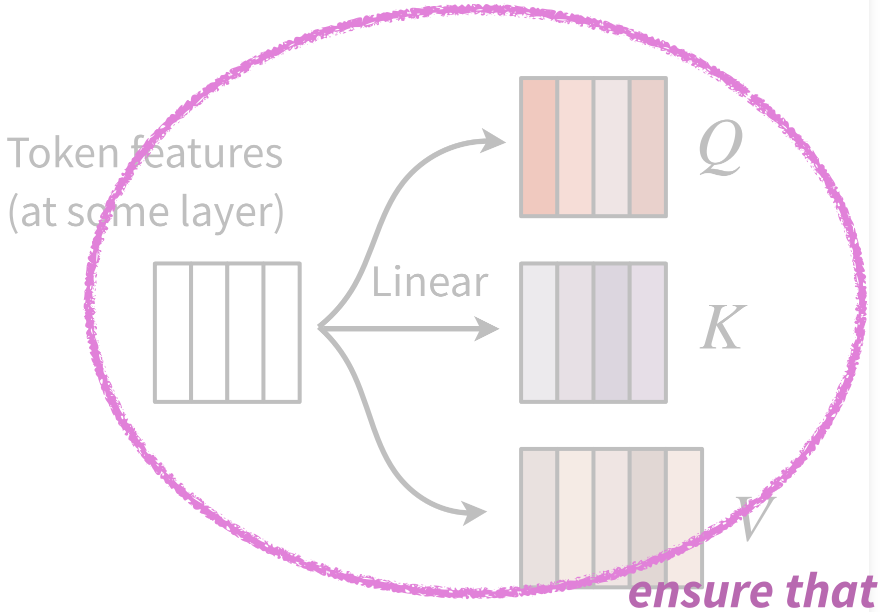
2. Calculate $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q^T K}{\sqrt{d}}\right)V$



The Recipe

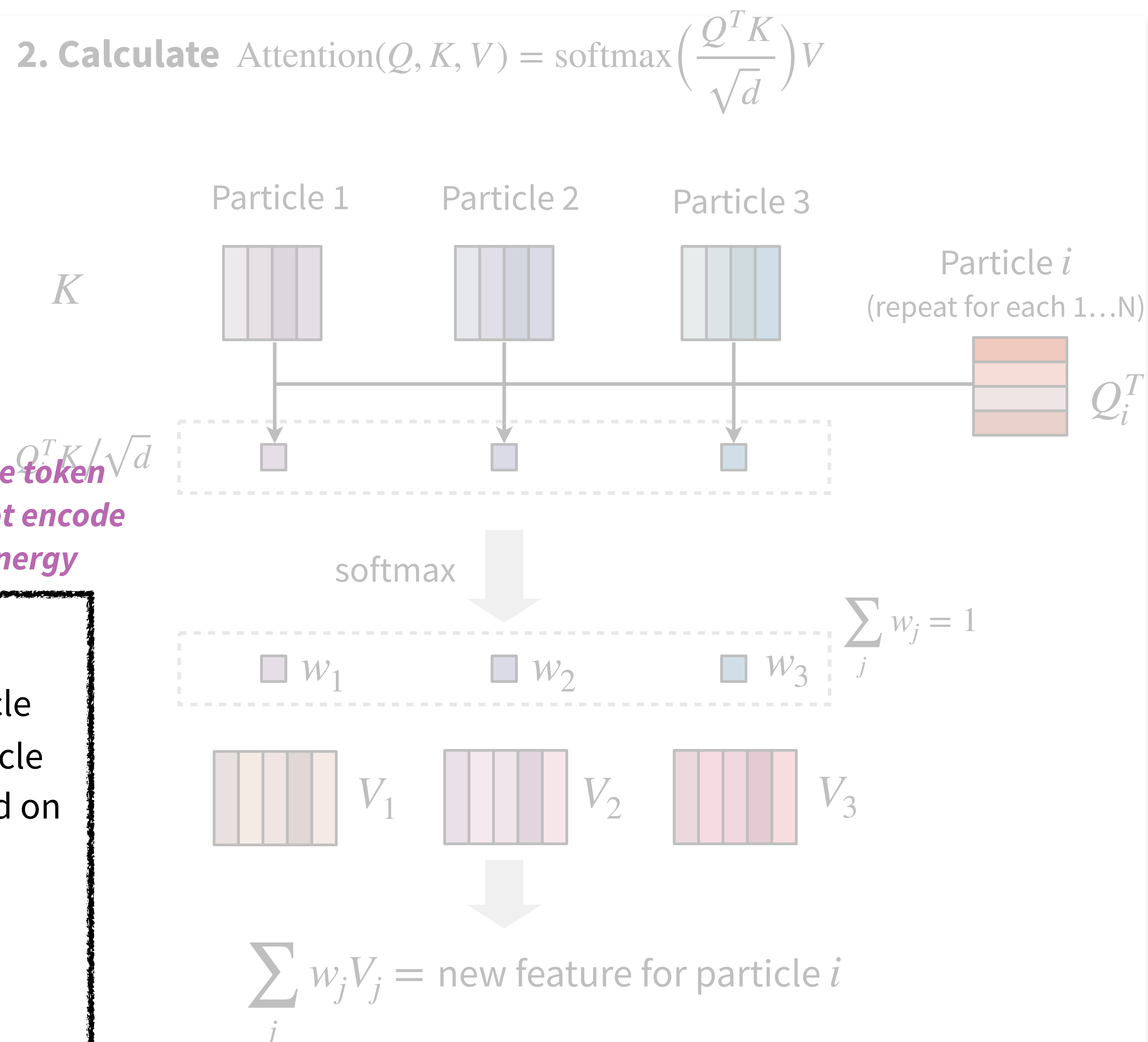
Revisiting dot-product attention

1. Derive Q, K, V



ensure that the token features do not encode particle's p_T/energy

2. Calculate $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q^T K}{\sqrt{d}}\right) V$

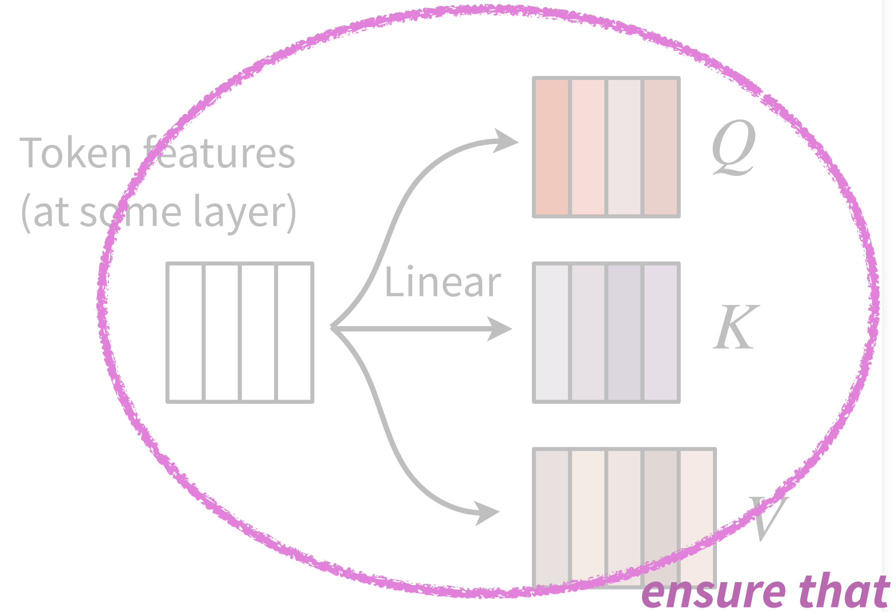


The Recipe

- Rule #1: before calculating every possible input, normalise each particle to have $E = 1$ (we assume zero particle mass). Then, calculate all input based on these normalised particles 4-vector

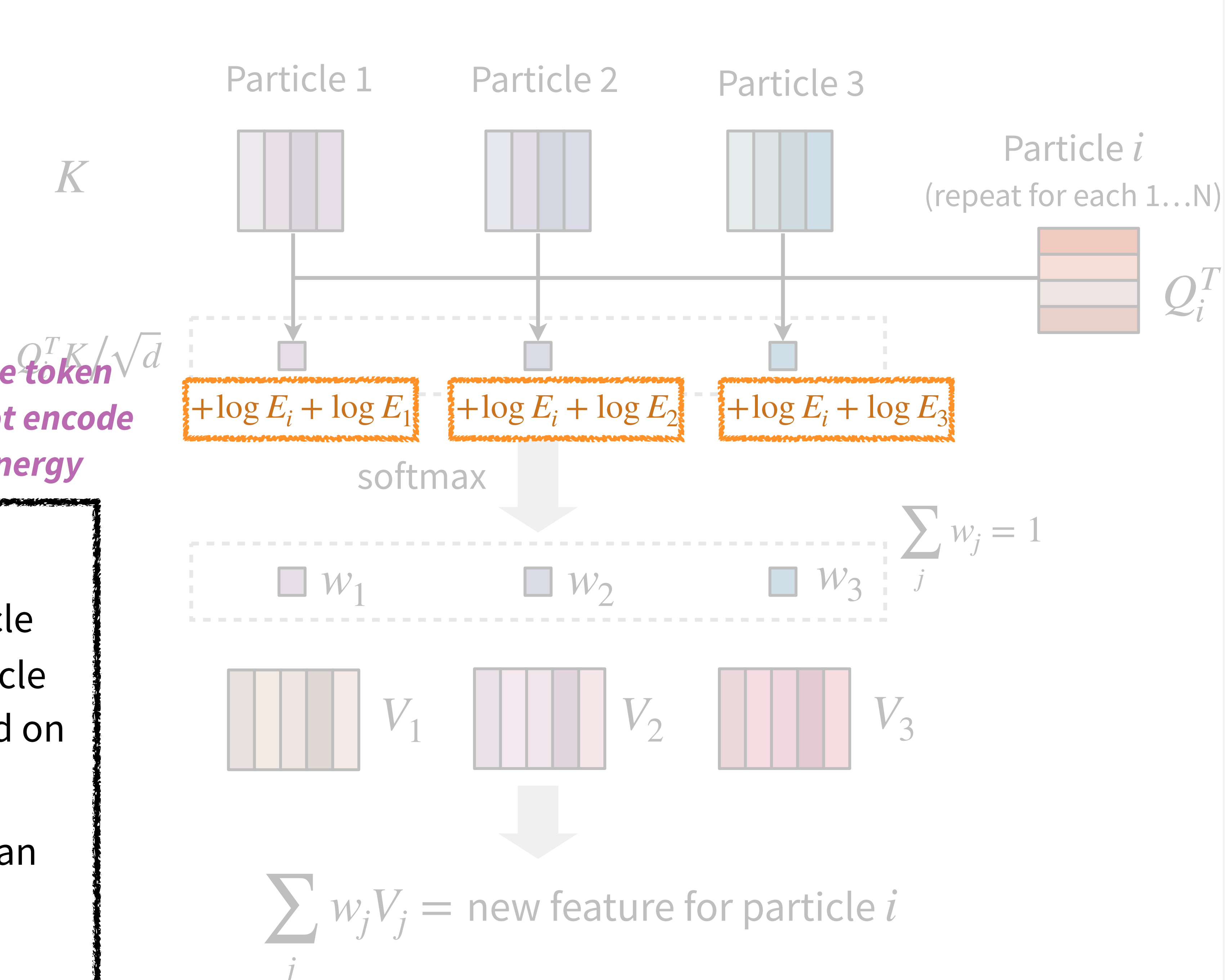
Revisiting dot-product attention

1. Derive Q, K, V



ensure that the token features do not encode particle's p_T/energy

2. Calculate $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q^T K}{\sqrt{d}}\right) V$

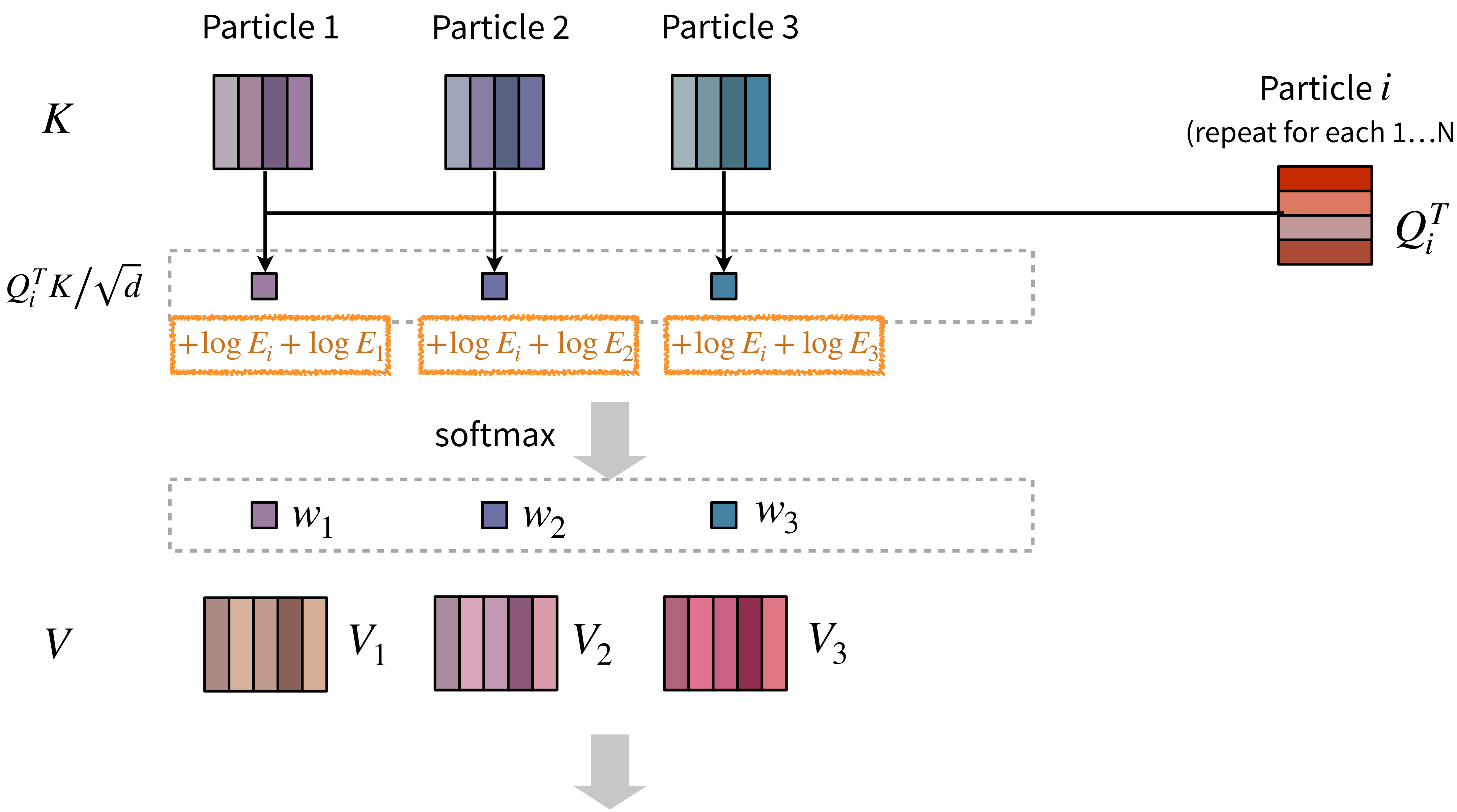


The Recipe

- Rule #1: before calculating every possible input, normalise each particle to have $E = 1$ (we assume zero particle mass). Then, calculate all input based on these normalised particles 4-vector
- Rule #2: for all attention blocks, add an attention additive bias $U_{ij} = \log E_i + \log E_j$

A figurative proof

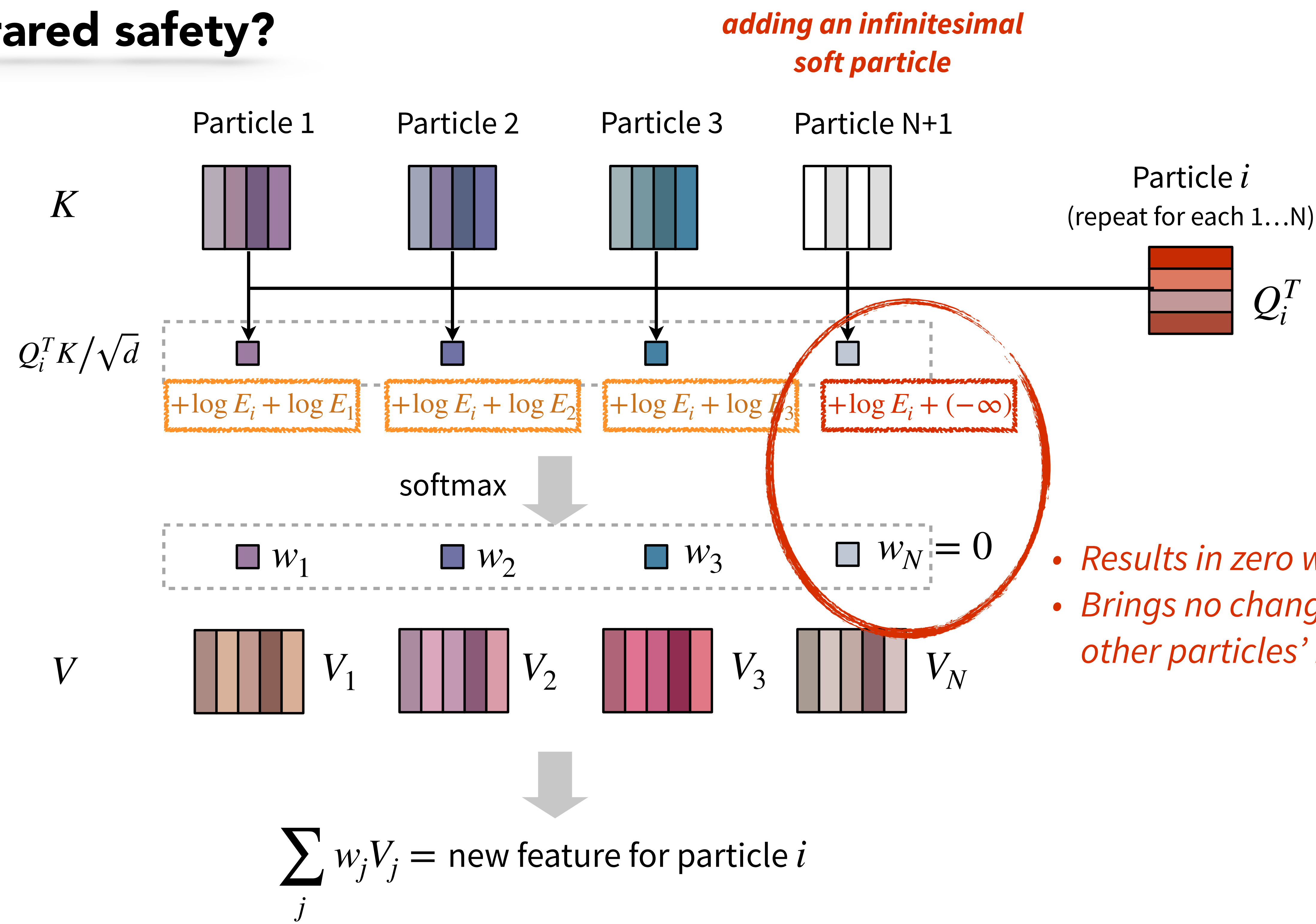
Infrared safety?



$$\sum_j w_j V_j = \text{new feature for particle } i$$

A figurative proof

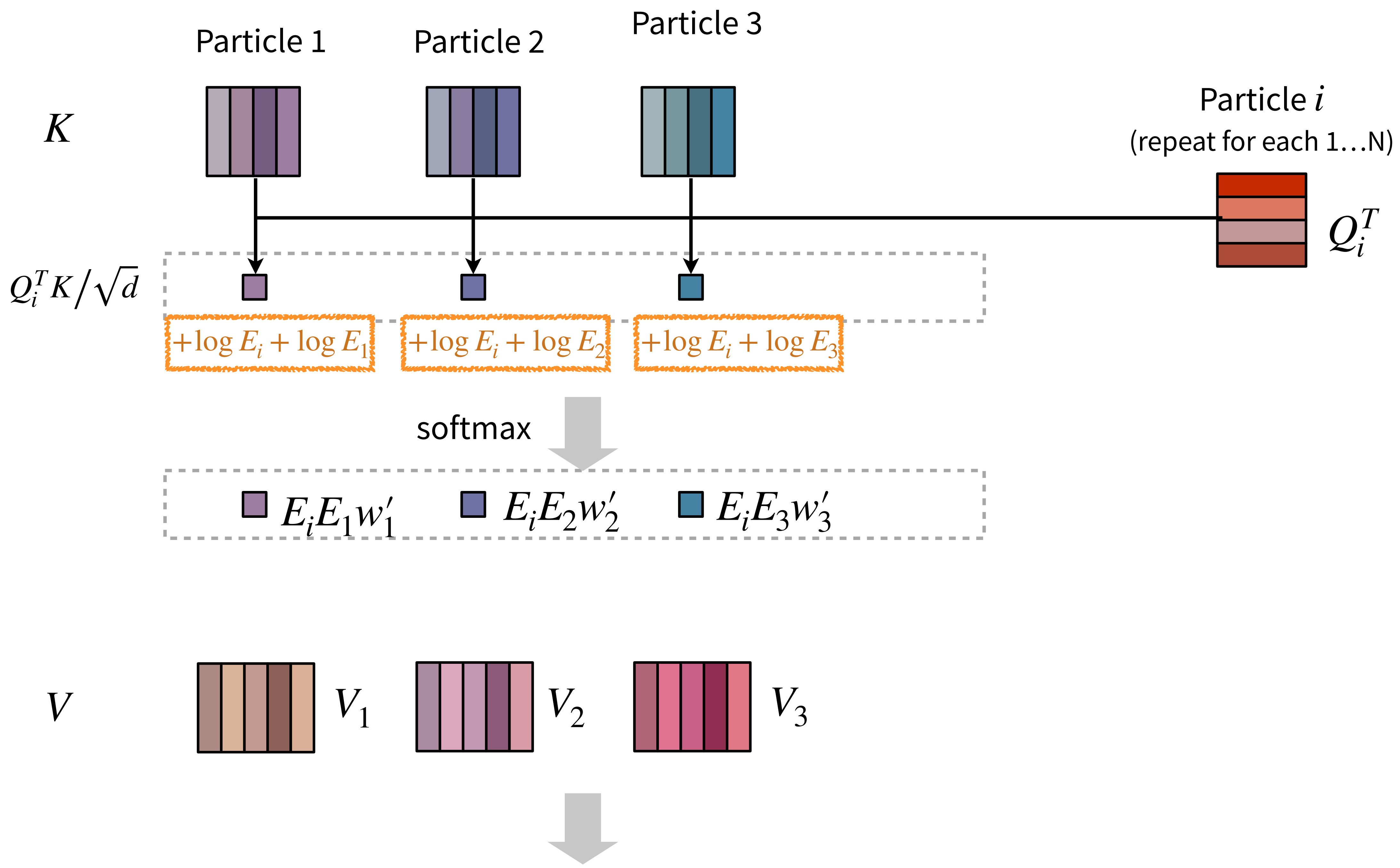
Infrared safety?



- *Results in zero weight*
- *Brings no change to the other particles' features*

A figurative proof

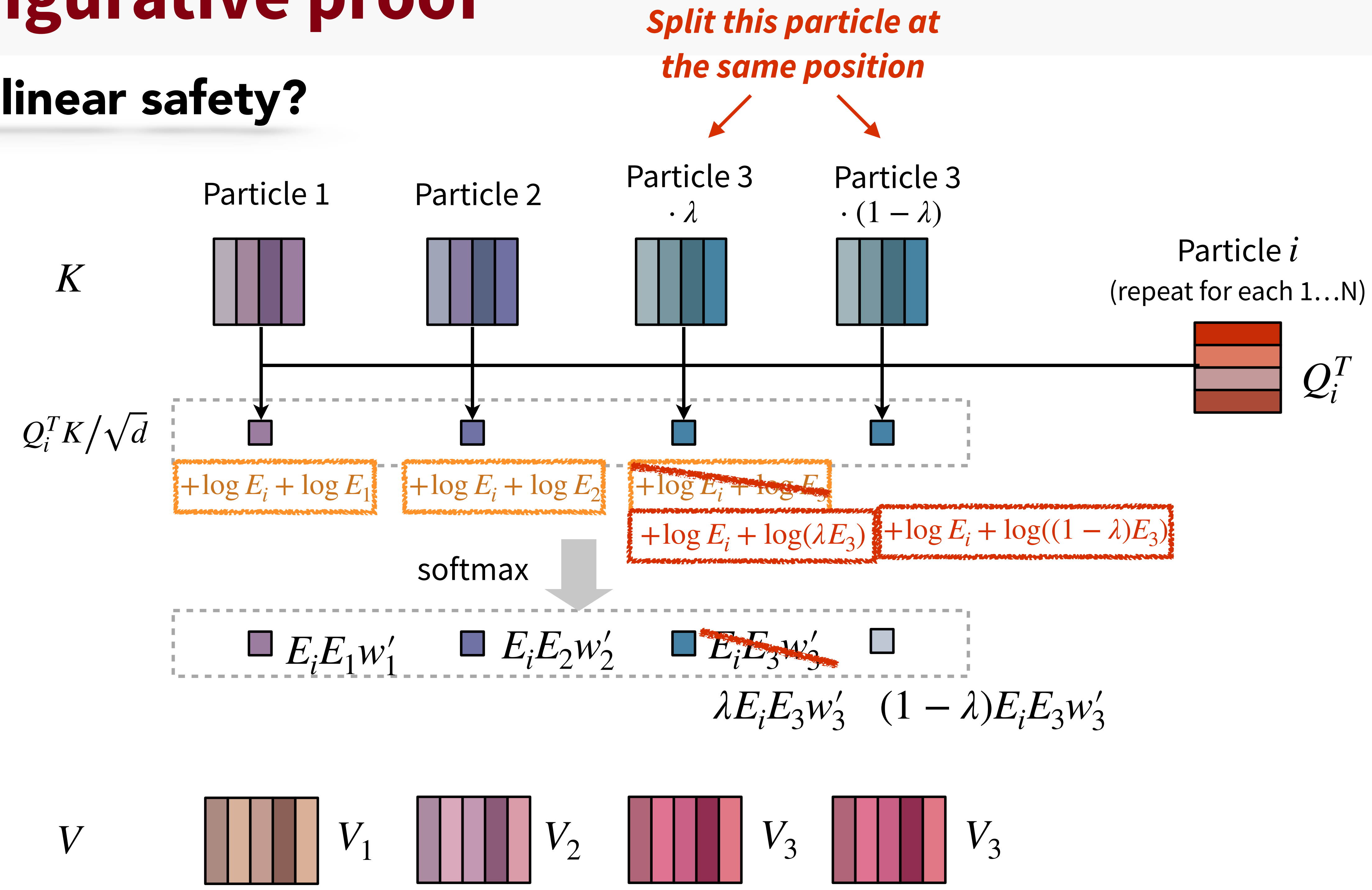
Collinear safety?



$$\sum_j w_j V_j = \text{new feature for particle } i$$

A figurative proof

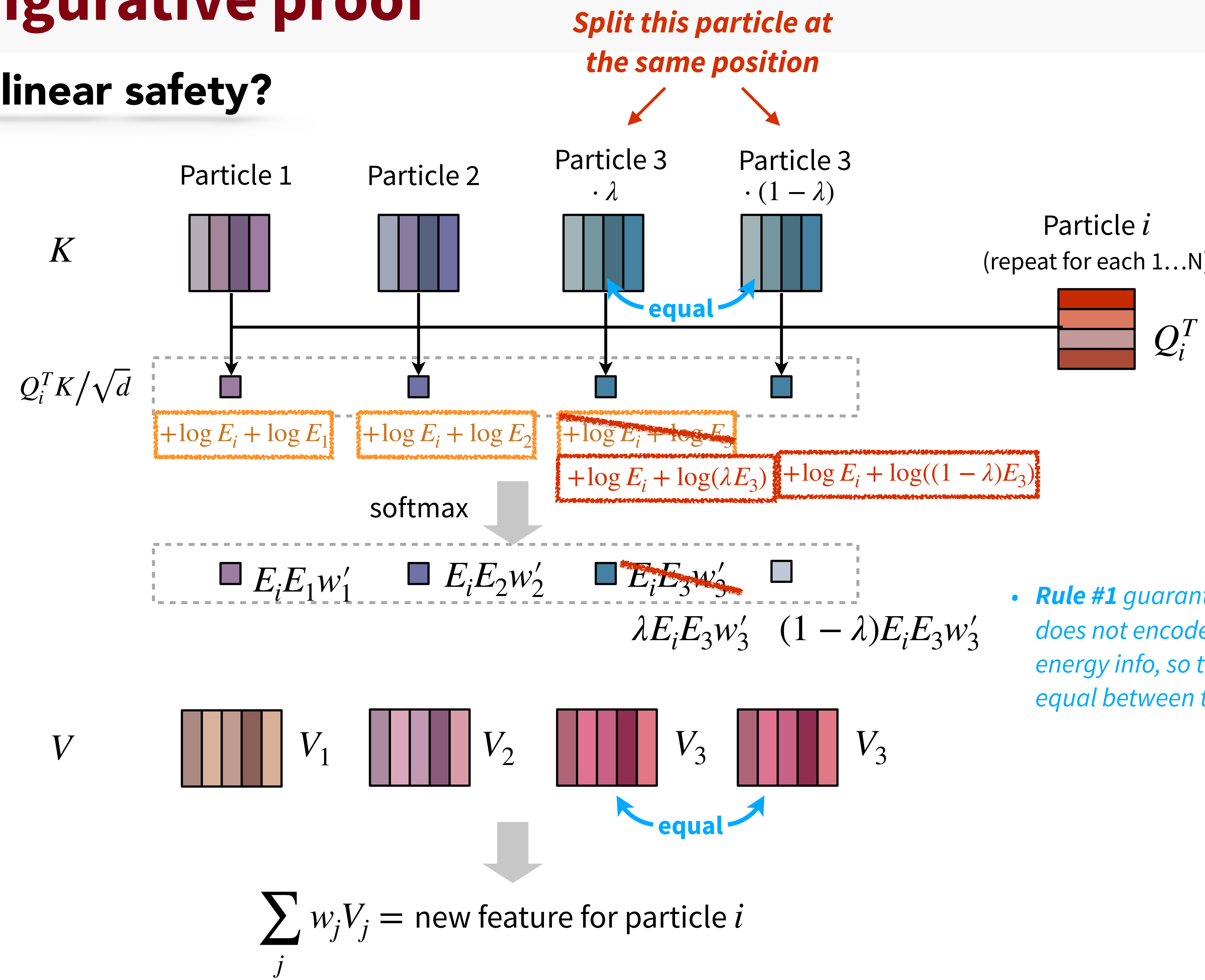
Collinear safety?



$$\sum_j w_j V_j = \text{new feature for particle } i$$

A figurative proof

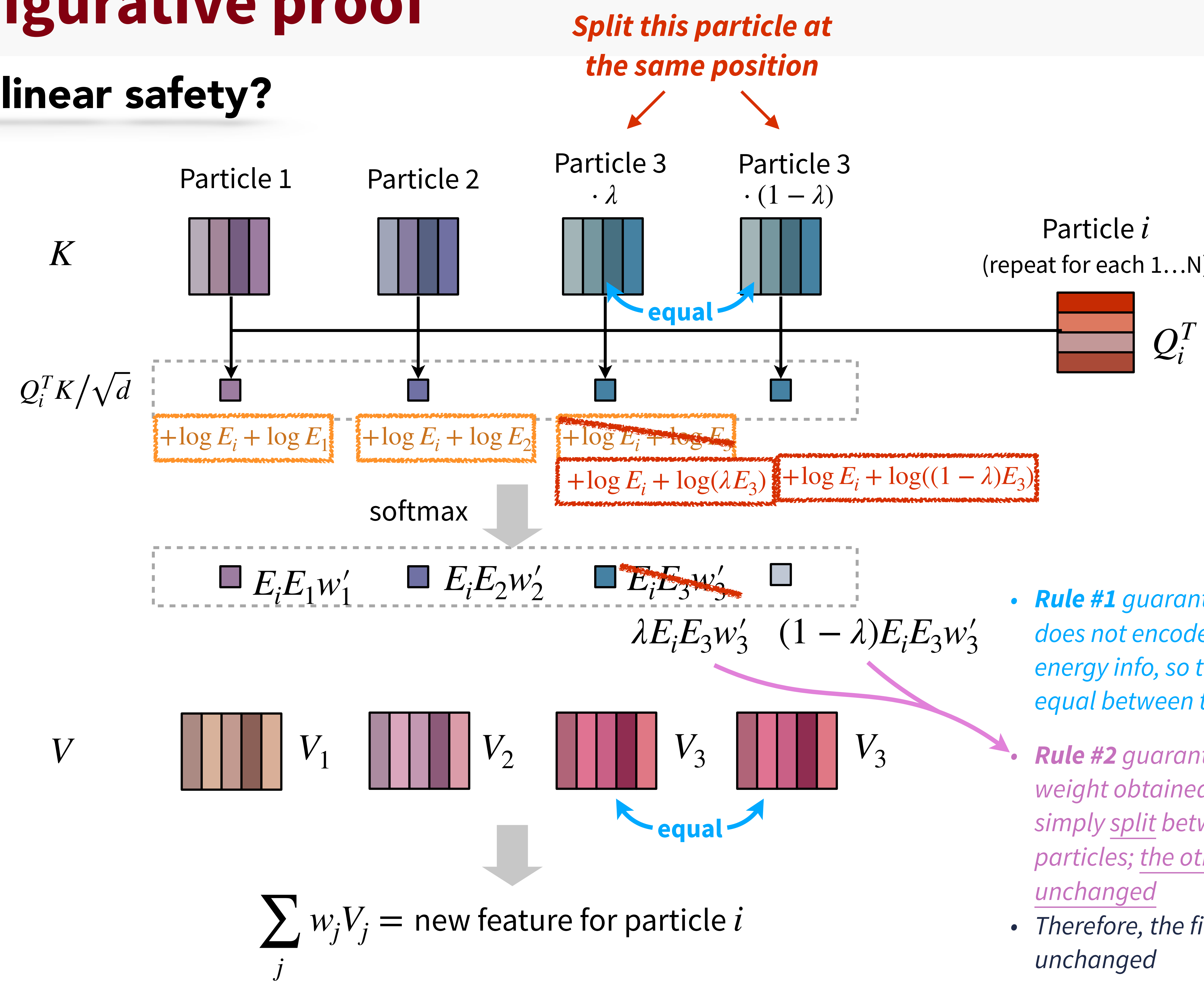
Collinear safety?



- ***Rule #1*** *guarantees that K, V does not encode the particle energy info, so they should be equal between two split particles*

A figurative proof

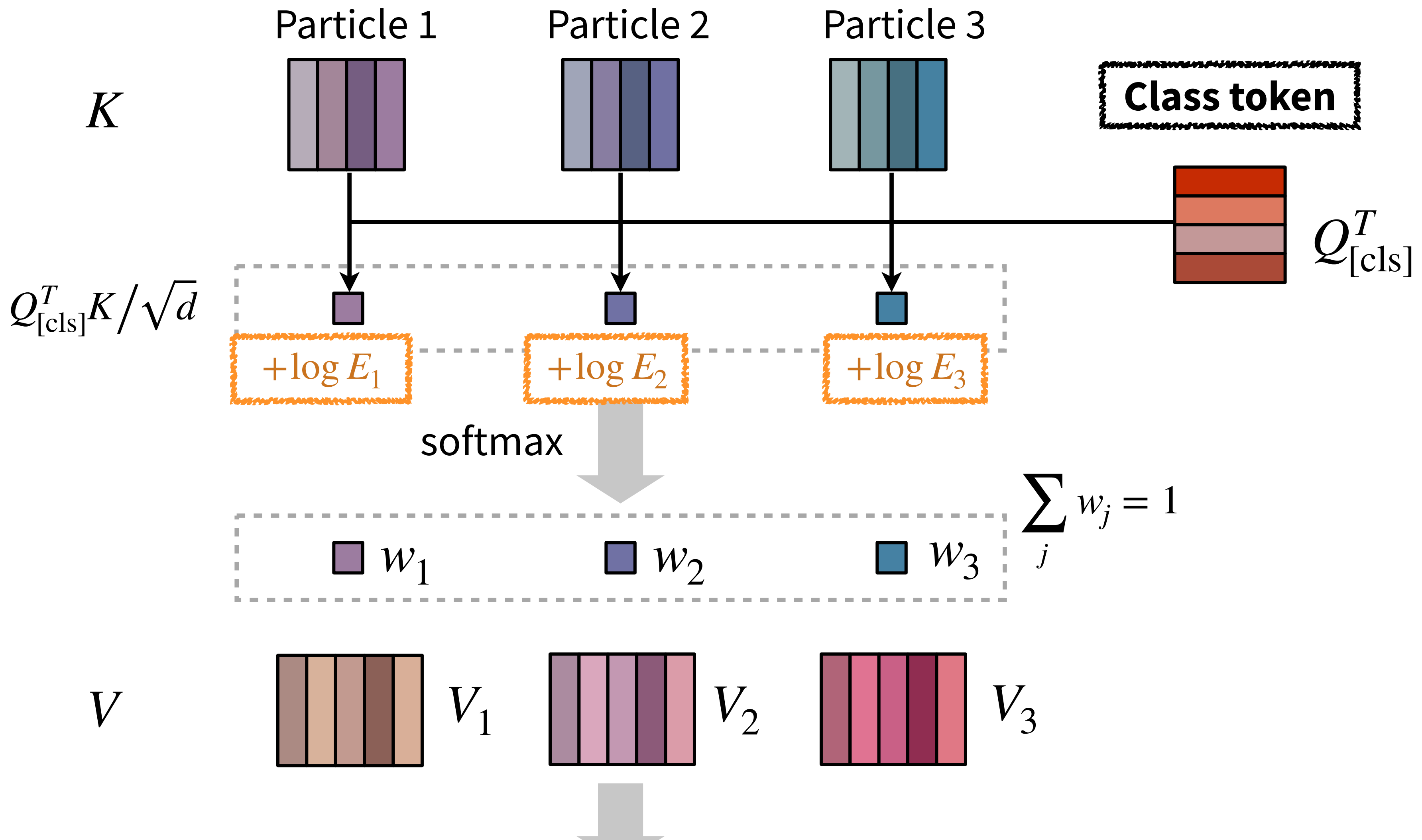
Collinear safety?



- ***Rule #1*** *guarantees that K, V does not encode the particle energy info, so they should be equal between two split particles*
- ***Rule #2*** *guarantees that the weight obtained after softmax is simply split between the two particles; the other weights are unchanged*
- *Therefore, the final result is unchanged*

About final aggregation

- For standard aggregation, we can only sum over all token features using energy weights
  - similar to the handle in $\mathrm{PELICAN}_{\mathrm{IRC}}$
- Or, if we use a class token to form the final classification outputs with one or more "class attention blocks", like ParT does, we just play the trick again
  - now adding $U_{[\mathrm{cls}],i} = \log E_i$ as attention bias, as there is only one query



$\sum_j w_j V_j =$ the aggregated feature, which is IRC safe

Intuitive understanding

- ➔ Why is this recipe a general one for all particle-based Transformers?
  - ❖ All intermediate neurons/features can be considered particle-based features (token features) or overall features
  - ❖ Rule #1 guarantees that **all token features do not encode energy** (so only have the geometric information of a particle)
    - ‣ in this sense, even the *pairwise features in ParT* are independent of particles' energies, as they are obtained from normalized-vector
  - ❖ Rule #2 modifies the attention block (the only inter-particle communicating process in a Transformer model) to guarantee that the above is always satisfied
  - ❖ Lastly, after aggregation, **all overall features should be IRC-safe variables**; hence the output scores

Training $\mathrm{ParT}_{\mathrm{IRC}}$

[H. Qu, C. Li, and S. Qian, 2202.03772]

➔ We use ParT as an example:

- Train an IRC-safe ParT ($\mathrm{ParT}_{\mathrm{IRC}}$) on JetClass
- only uses 10M jets for training
- two configurations
  - **$\mathrm{ParT}_{\mathrm{IRC}}$ (full)**: use full particle input
  - **$\mathrm{ParT}_{\mathrm{IRC}}$ (kin)**: only *using particles' kinematics input* (no PID, charge, IP info)
- model: ParT backbone + IRC recipe

	All classes	
	Accuracy	AUC
ParticleNet (2 M)	0.828	0.9820
ParticleNet (10 M)	0.837	0.9837
ParticleNet (100 M)	0.844	0.9849
ParT (2 M)	0.836	0.9834
ParT (10 M)	0.850	0.9860
ParT (100 M)	0.861	0.9877

Preliminary

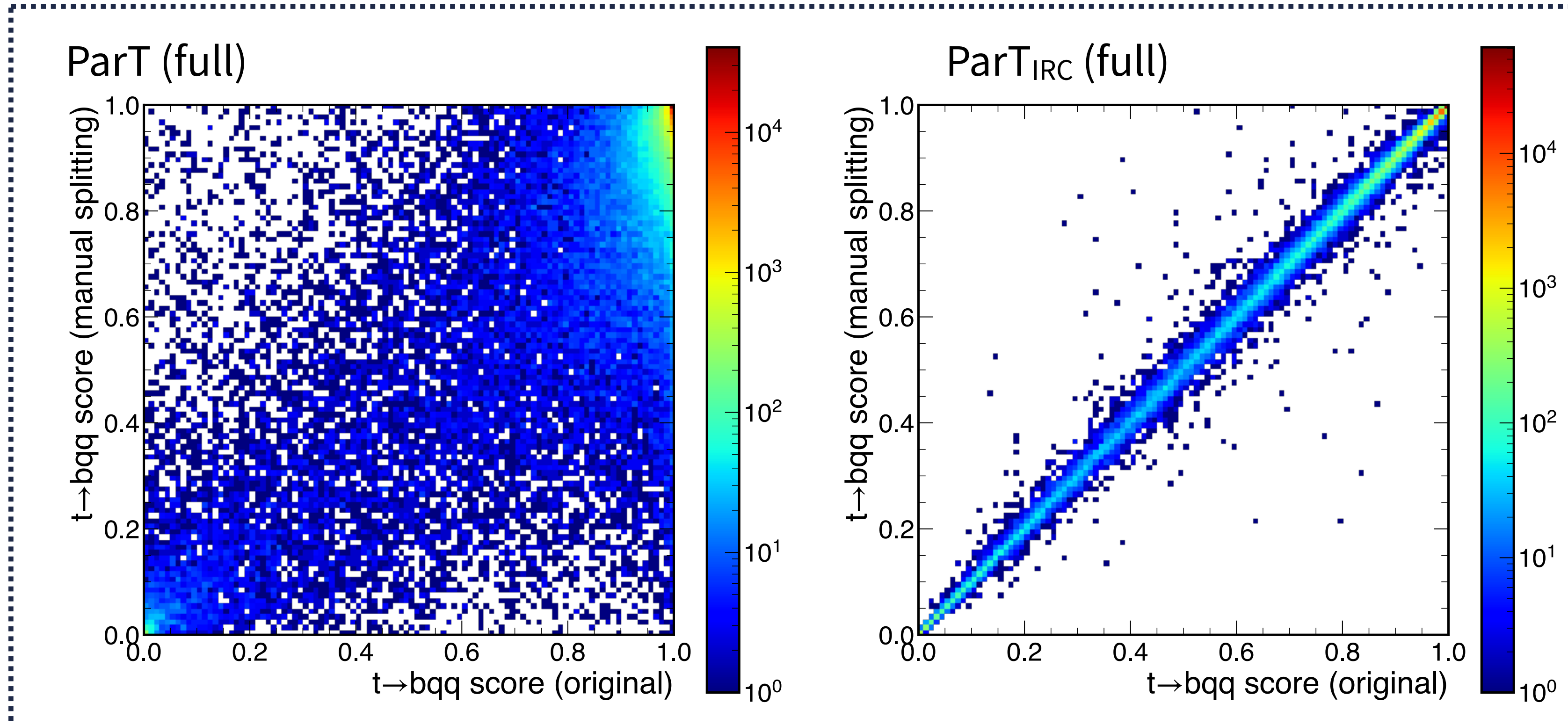
ParT (10M, full)	0.850	0.9860
$\mathrm{ParT}_{\mathrm{IRC}}$ (10M, full)	0.847	0.9856
ParT (10M, kin)	0.738	0.9635
$\mathrm{ParT}_{\mathrm{IRC}}$ (10M, kin)	0.729	0.9614

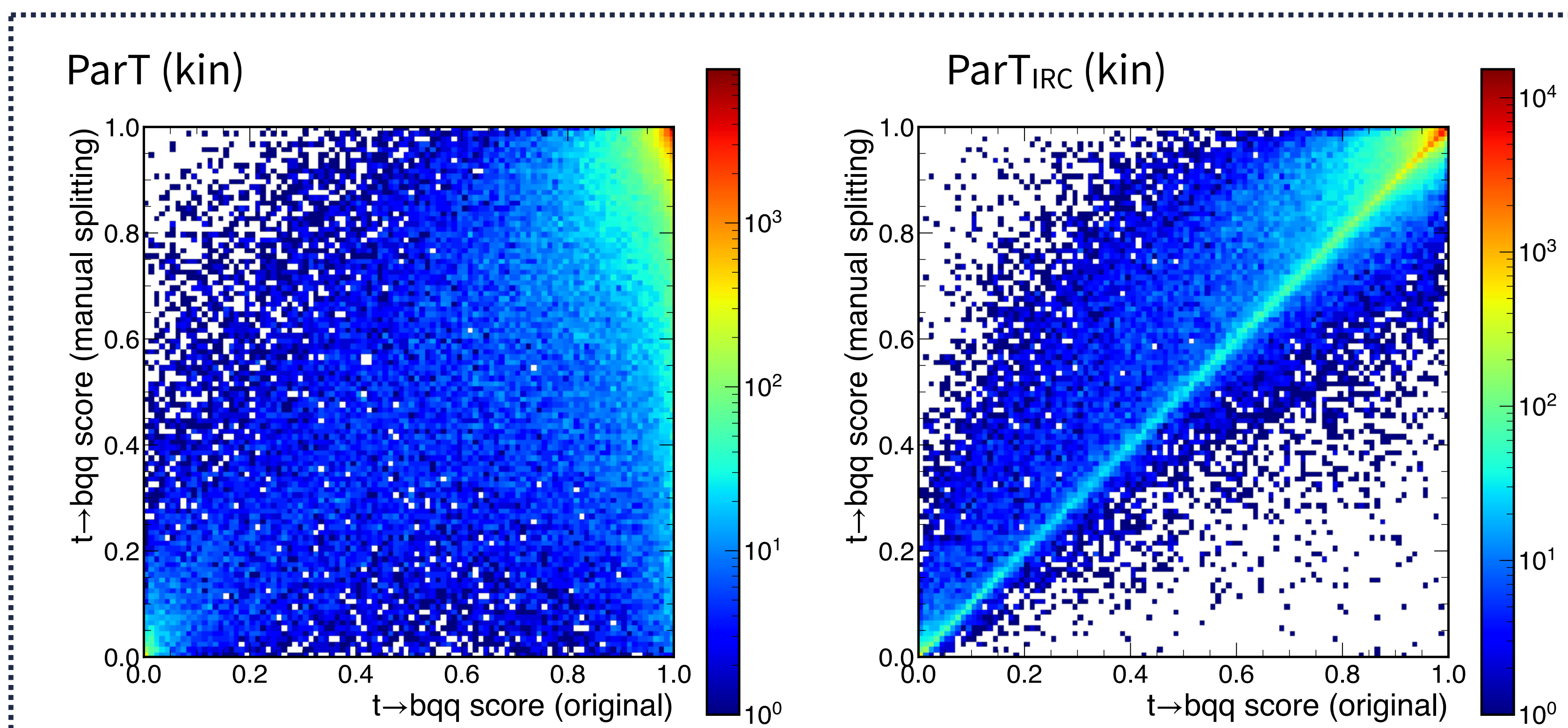
No significant drop in performance

Testing IRC safety

note: for models requiring full particle input, we make a non-physical assumption that the other properties of two split particles are the same

➔ Test the IRC-safe property
- ❖ producing "fake jets": manually split 30% of particles with arbitrary λ
- ❖ comparing the output scores with the original jet





- *prove that IRC safety is built into the ParT$_{IRC}$ network*
- *the deviation from the $y = x$ line indicates the numerical error*
- *see larger numerical uncertainties for the kin-only model*

Top tagging and quark/gluon tagging

➔ Also evaluate the $\mathrm{ParT_{IRC}}$ on two tagging benchmarks

[A. Bogatskiy et al. 2307.16506]

- $\mathrm{PELICAN_{IRC}}$: a recently introduced IRC-safe modification of PELICAN (see previous talk)
- by network design philosophy, our Transformer's recipe, $\mathrm{PELICAN_{IRC}}$, EMPN, EFN are very similar, i.e. not embedding energy into particle features, but using it as a "particle weight"

Top tagging benchmark

Architecture	Accuracy	AUC	$1/\epsilon_B$	# Params
TopoDNN[48]	0.916	0.972	382± 5	59k
EFN[24]	0.927	0.979	729 ± 13	82k
LGN[25]	0.929(1)	0.964(14)	424 ± 82	4.5k
BIP(XGBoost)[49]	0.929	0.978	600 ± 47	312
EFP[18]	0.932	0.980	384	1k
BIP(MLP)[49]	0.931	0.981	853 ± 68	4k
PFN[24]	0.932	0.982	891 ± 18	82k
ResNeXt[8]	0.936	0.984	1122 ± 47	1.46M
ParticleNet[50]	0.938	0.985	1298 ± 46	498k
ParT[35]	0.940	0.9858	1602 ± 81	2.1M
LorentzNet[26]	0.942	0.9868	2195 ± 173	220k
PELICAN	0.9426(2)	0.9870(1)	2250 ± 75	208k
$\mathrm{PELICAN_{IRC}}$	0.9406(2)	0.9844(11)	1711 ± 208	208k
$\mathrm{ParT_{IRC}}$ (kin)	0.9350(6)	0.9836(4)	1054 ± 67	2.1 M

Preliminary

Quark/gluon tagging benchmark

Architecture	Accuracy	AUC	$1/\epsilon_B$ ($\epsilon_S = 0.3$)	$1/\epsilon_B$ ($\epsilon_S = 0.5$)	# Params
Not IRC-safe, w/ PID					
PFN-ID[24]	–	0.9052(7)	–	37.4 ± 0.7	82k
ParticleNet-ID[50]	0.840	0.9116	98.6 ± 1.3	39.8 ± 0.2	498k
ABCNet[26]	0.840	0.9126	118.2 ± 1.5	–	230k
LorentzNet[26]	0.844	0.9156	110.2 ± 1.3	42.4 ± 0.4	220k
$\mathrm{ParT_{full}}$ [35]	0.849	0.9203	129.5 ± 0.9	47.9 ± 0.5	2.1M
$\mathrm{PELICAN_{PID}}$	0.8555(2)	0.9247(3)	134.8 ± 1.8	51.3 ± 0.7	211k
Not IRC-safe, w/o PID					
PFN[24]	–	0.8911(8)	–	30.8 ± 0.4	82k
ParticleNet[50]	0.828	0.9014	85.4	33.7	498k
PELICAN	0.8342(2)	0.9059(8)	88.9 ± 0.5	36.0 ± 0.2	209k
IRC-safe					
EFN[24]	–	0.8824(5)	–	28.6 ± 0.3	82k
EFP[18]	–	0.8919	–	29.7	1k
EMPN[55]	–	0.8932(6)	–	30.8 ± 0.2	~110k
$\mathrm{PELICAN_{IRC}}$	0.8299(3)	0.8955(18)	85.7 ± 1.2	33.8 ± 0.2	209k
$\mathrm{ParT_{IRC}}$ (kin)	0.8260(6)	**0.9008(2)**	**87.8 ± 1.3**	**34.0 ± 0.2**	2.1 M

Note: numerical error prevents us from achieving "perfect IRC safety", which can be more severe in Transformers. → requires deeper investigation!

Summary

- ➔ We present a recipe that can build in IRC safety into all particle-based Transformers
  - ❖ our trick is a general one
    - Transformer tokens can be particles or analysis-level objects (jet/lepton)
    - used in Transformers with various tasks (tagging/regression/point cloud generation)..
  - ❖ the concept is to hack into the "attention weight"; easy to implement using PyTorch `nn.MultiheadAttention`
  - ❖ the essence of our recipe is still to use energy as particle weights (similar to EFN, EMPN, $\mathrm{PELICAN}_{\mathrm{IRC}}$)
- ➔ These slides show preliminary results; more work ongoing
  - ❖ code and paper will be available soon
  - ❖ some further directions for this study (next slides)

Outlook

- The $\mathrm{ParT}_{\mathrm{IRC}}$ (along with $\mathrm{PELICAN}_{\mathrm{IRC}}$) model can facilitate further theoretical study on the IRC-safe observables
  - e.g. can we interpret its good performance using well-established IRC theory, such as expanding it with EFPs?
- Understand the numerical stability of the IRC-safe model
  - e.g. does any special embedding of node features or pairwise features impact the stability?
- Next-up for $\mathrm{ParT}_{\mathrm{IRC}}$/ParT model design?
  - helpful to boost into the jet's C.O.M frame? → should bring more inductive bias as it will make the pairwise normalized-vector feature an invariance
  - helpful to introduce more operations to process pairwise masses (draw from PELICAN's experience)? → better-learned representation