DEEP(BOOSTED)JET:
Boosted Jet Identification Using Particle-Level Convolutional Neural Networks

Huilin Qu

on behalf of the CMS collaboration

University of California, Santa Barbara

MACHINE LEARNING FOR JET PHYSICS

DECEMBER 11, 2017

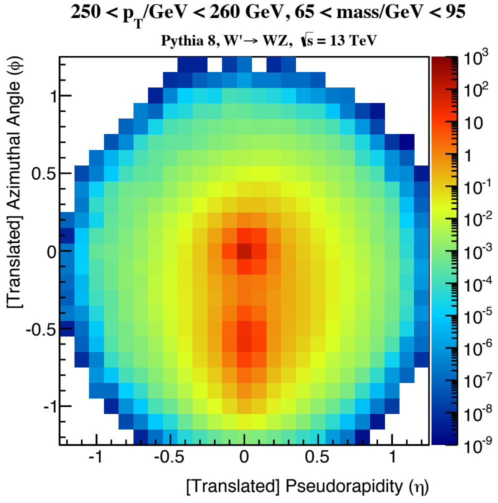
Introduction

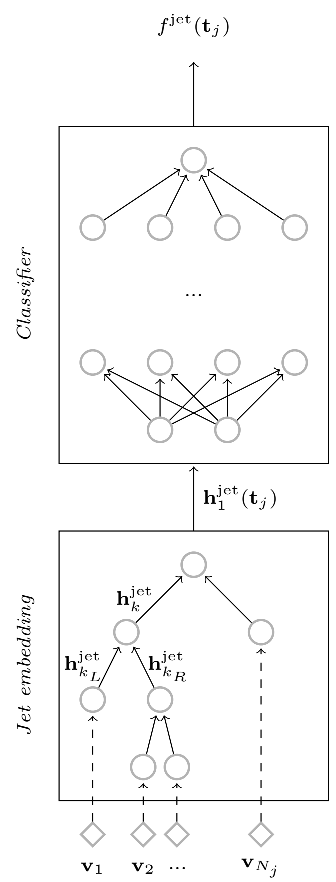
- Reconstruction and identification of boosted heavy particles (top/W/Z/Higgs) can provide powerful handles for new physics searches at the LHC
- Of particular interest are their hadronic decays
  - highly collimated decay products: can be clustered to form a single large-radius jet
  - unique substructure: distinguishable from jets initiated by a single quark or gluon
- There has been a lot of development in jet substructure
  - better understanding the properties of these boosted jets
  - improvement in identification performance
- Also a growing interest in applying machine learning techniques
  - especially using deep neural networks (DNN) and “raw” inputs

DEEP LEARNING APPROACHES

- Two categories of deep learning approaches for jet tagging:
  - Based on jet image:
    - calorimeter as a camera and jet as a 2D image
    - can apply techniques used for image recognition (e.g., convolutional neural networks (CNN))
    - however, jet images are much more sparse than normal images
    - also difficult to embed information from other sub-detectors (e.g., tracking)
  - Based on particle sequence:
    - jet as a sequence of its constituent particles
    - can apply techniques used for natural language processing (e.g., recurrent neural networks, 1D CNN, etc.)
    - straightforward to include as much information as possible for each particle
    - more suitable for exploiting the full potential of the CMS detector

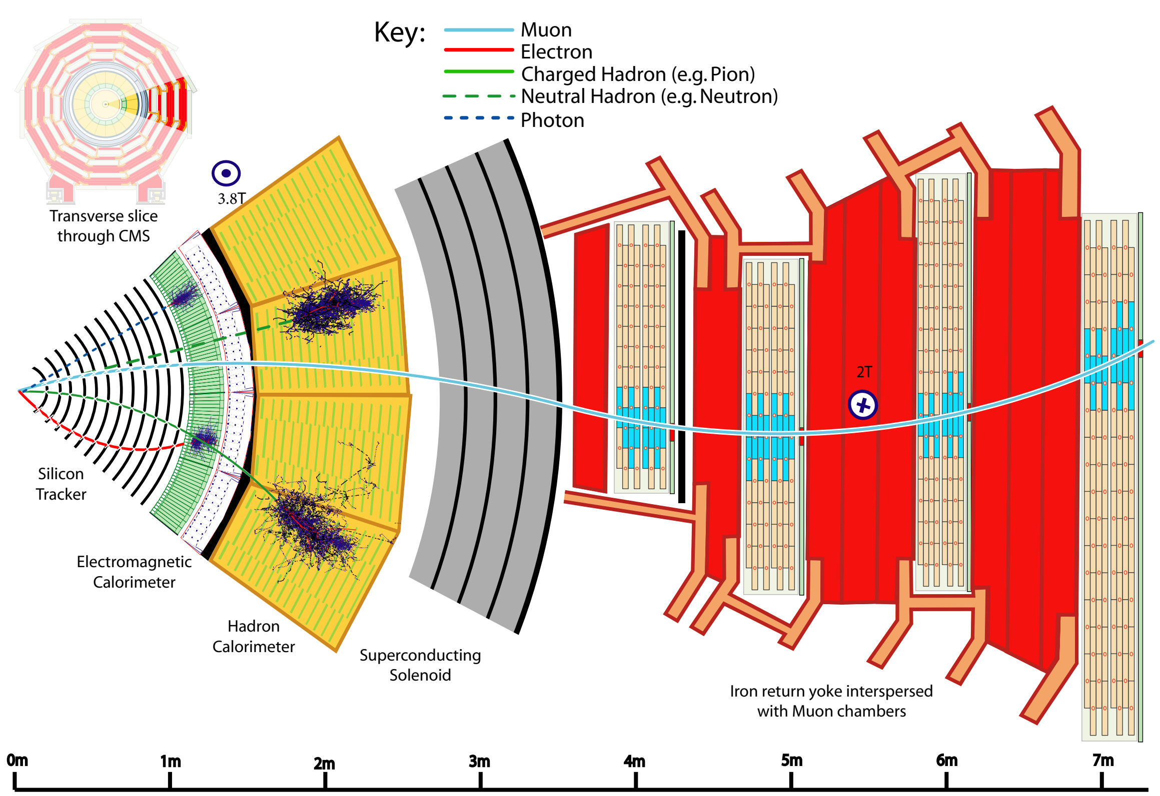
arXiv:1511.05190





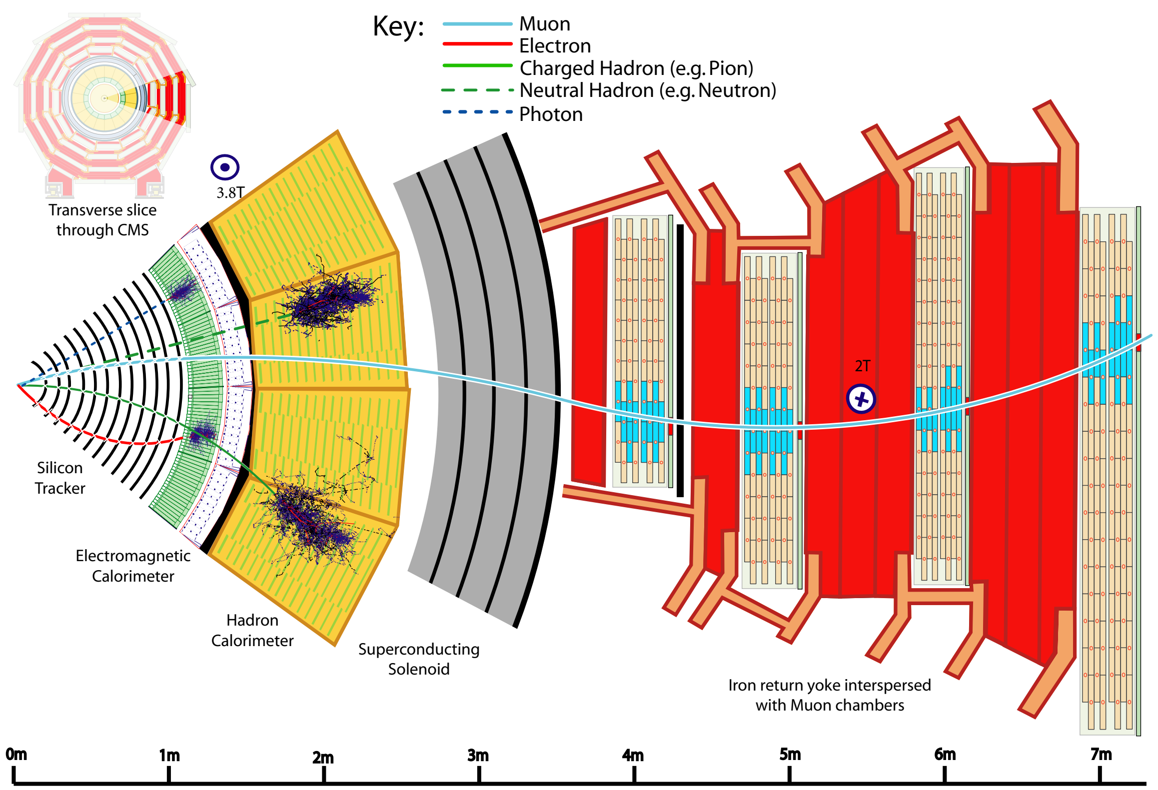
arXiv:1702.00748

Particle-Flow Jets in CMS



- In CMS, jets are clustered from particles reconstructed with the CMS particle-flow algorithm, which combines information from all sub-detectors
- Rich information for each particle:
    - energy/momentum
    - particle category
    - trajectories of charged particles
        - vertex association
        - displacement from the primary vertex
        - reconstruction quality
        - …

PARTICLE-FLOW JETS IN CMS

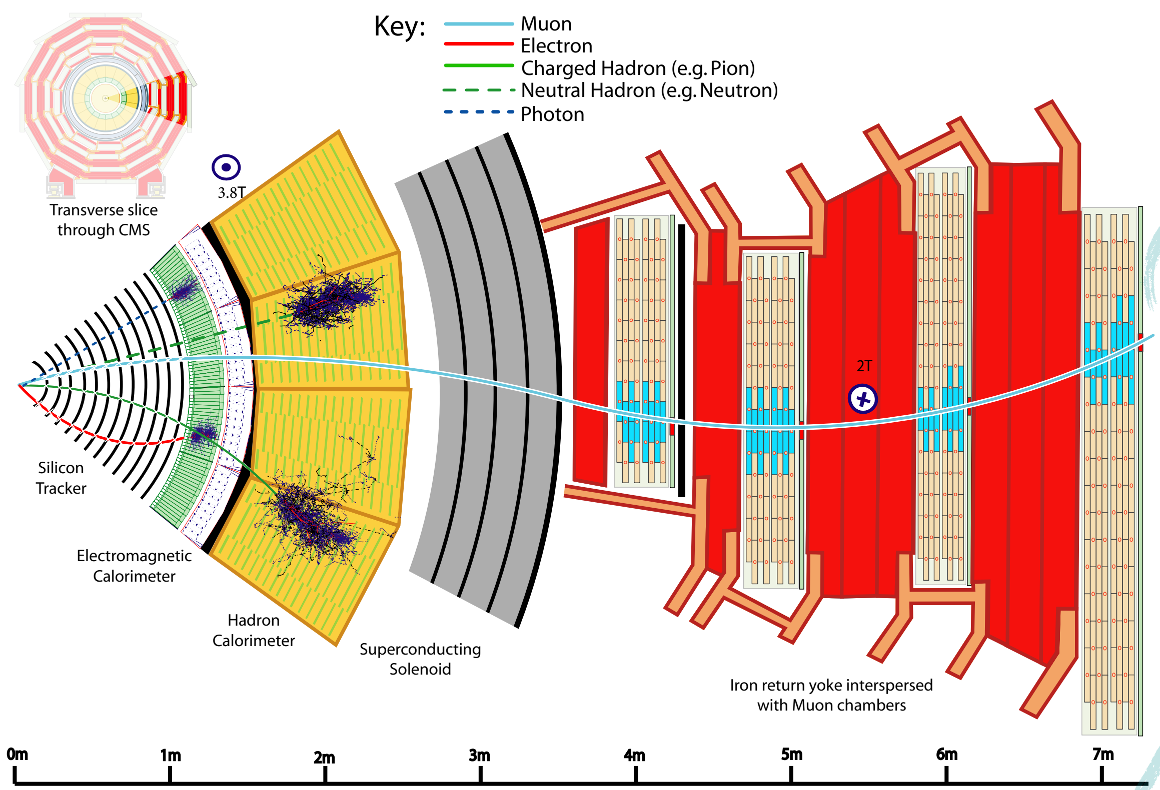


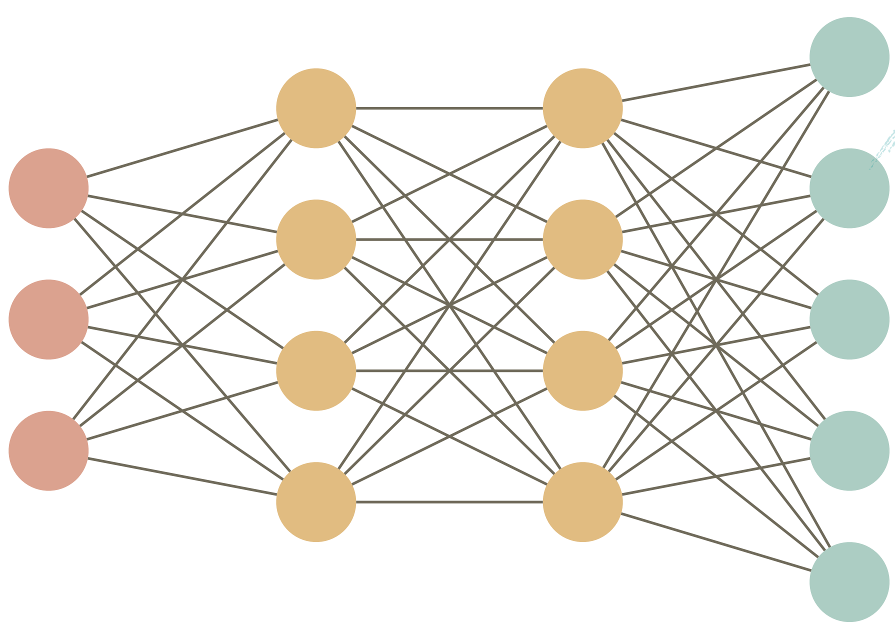
- In CMS, jets are clustered from particles reconstructed with the CMS particle-flow algorithm, which combines information from all sub-detectors
- Rich information for each particle:
  - energy/momentum
  - particle category
  - trajectories of charged particles
    - vertex association
    - displacement from the primary vertex
    - reconstruction quality
    - ...

Inputs for jet substructure

- *Traditionally well exploited for jet flavor (i.e., b- or c-)tagging*
- *Not so well exploited for boosted jet tagging, but should also be powerful*
  - *especially for top(→bW) and Higgs(→bb)*

PARTICLE-FLOW + DEEP LEARNING





- *By treating a jet as a sequence of particles and using deep learning, we can exploit jet substructure (from energy/momentum measurement) and flavor information (from tracking) in one go and improve boosted jet tagging performance*
- Rich information for each particle:
  - energy/momentum
  - particle category
  - trajectories of charged particles
    - vertex association
    - displacement from the primary vertex
    - reconstruction quality
    - ...

INPUTS

- Use jet constituent particles from particle-flow reconstruction directly as inputs to the machine learning algorithm
  - the goal is to get as much information as possible on both substructure and flavor

Inclusive particles

- Up to 100(*) particles (charge+neutral)
- Sorted in descending p_T order
- 10 features per particle:
  - basic kinematic variables

Substructure

Flavour

Charged particles

- Up to 60(*) charged particles
- Sorted in descending impact parameter significance order
- 30 features per particle:
  - track kinematics and properties (displacement, quality, etc.)

Secondary vertices (SV)

- Up to 5(*) SV (inside jet cone)
- Sorted in descending displacement significance order
- 14 features per SV:
  - SV kinematics and properties (displacement, quality, etc.)

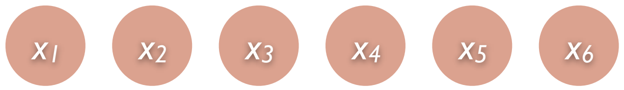
(*) Number chosen to include all constituents for ≥ 90% of the jets. Use zero padding if needed.

- In total: 100*10 + 60*30 + 5*14 = 2870 inputs per jet
  - need a ML algorithm that can process such inputs both effectively and efficiently

Particle-level CNN

- Particle-level CNN (P-CNN)
  - one dimensional CNN over a sequence of particles

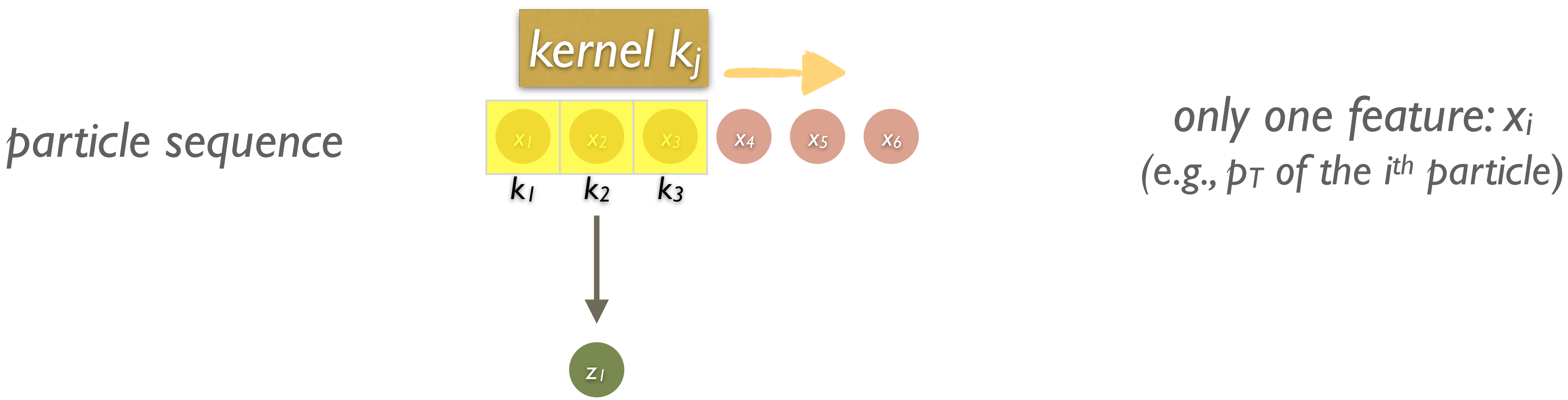
particle sequence



only one feature: x_i
(e.g., p_T *of the* i^{th} *particle)*

Particle-level CNN

- Particle-level CNN (P-CNN)
  - one dimensional CNN over a sequence of particles

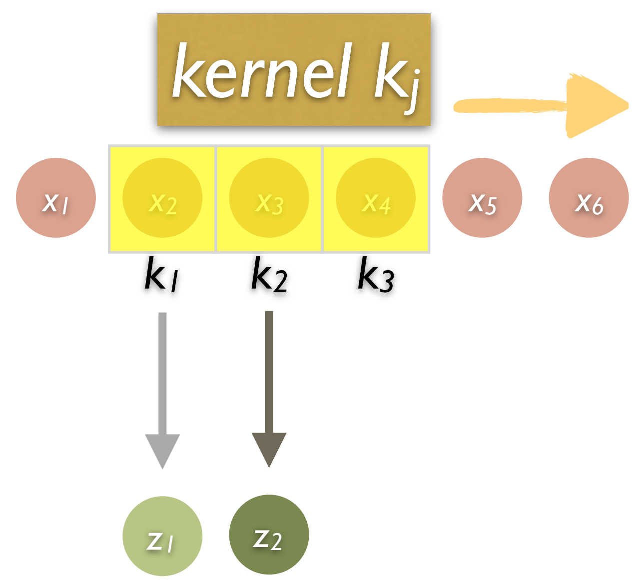


$z_1 = k_1 * x_1 + k_2 * x_2 + k_3 * x_3$

Particle-level CNN

- Particle-level CNN (P-CNN)
  - one dimensional CNN over a sequence of particles

particle sequence



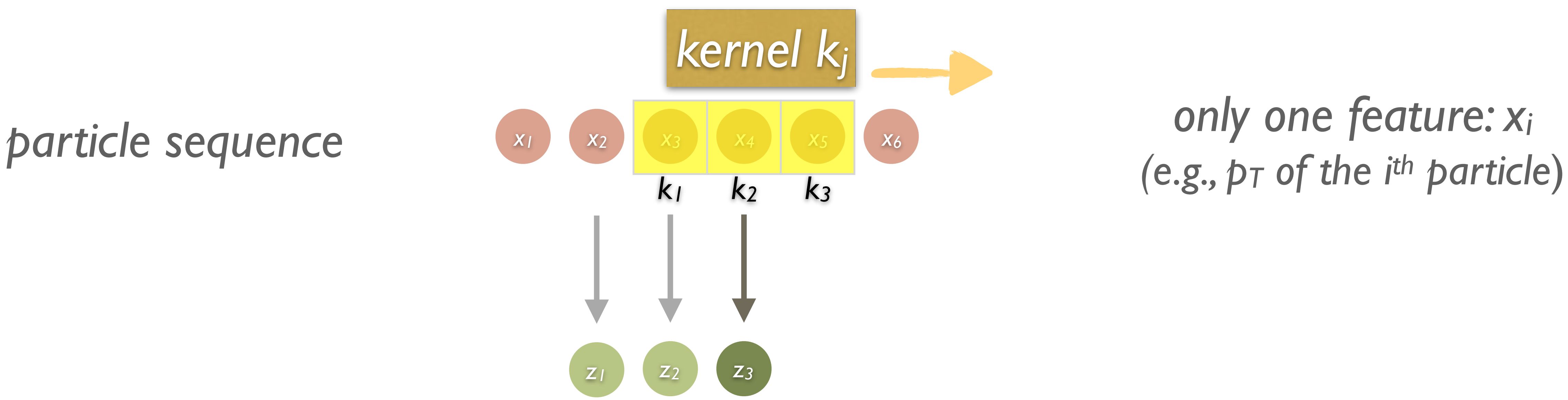
only one feature: x_i
(e.g., p_T of the i^{th} particle)

$z_1=k_1*x_1+k_2*x_2+k_3*x_3$

$z_2=k_1*x_2+k_2*x_3+k_3*x_4$

PARTICLE-LEVEL CNN

- Particle-level CNN (P-CNN)
  - one dimensional CNN over a sequence of particles



$z_1=k_1*x_1+k_2*x_2+k_3*x_3$

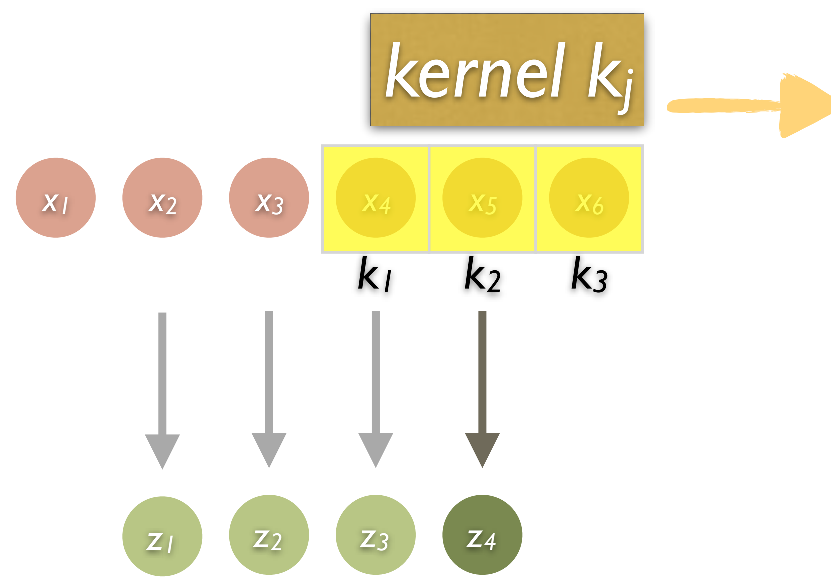
$z_2=k_1*x_2+k_2*x_3+k_3*x_4$

$z_3=k_1*x_3+k_2*x_4+k_3*x_5$

Particle-level CNN

- Particle-level CNN (P-CNN)
  - one dimensional CNN over a sequence of particles

particle sequence



only one feature: x_i
(e.g., p_T of the i^{th} particle)

$z_1=k_1*x_1+k_2*x_2+k_3*x_3$

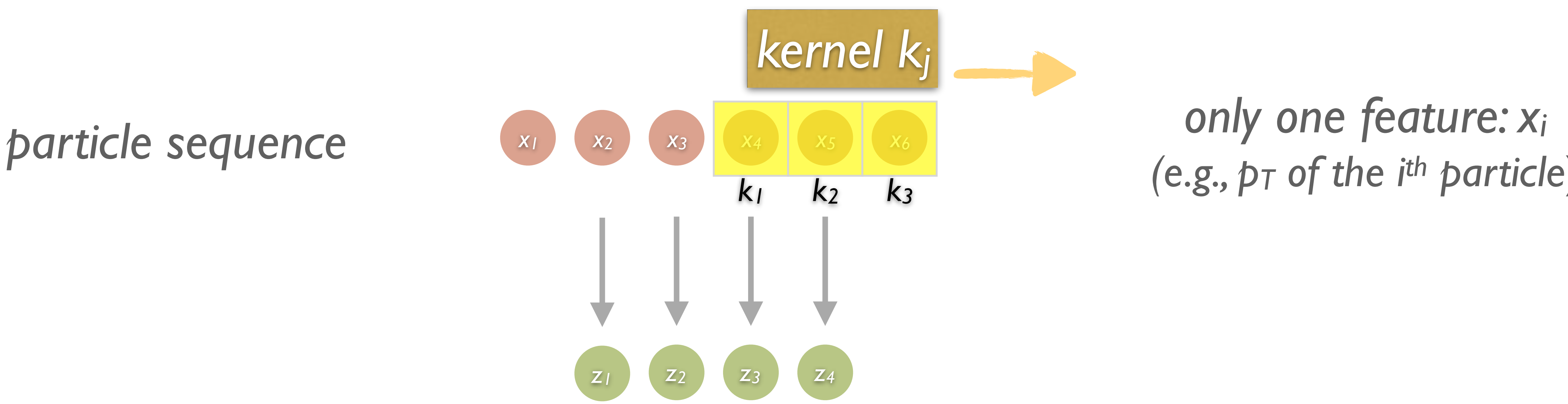
$z_2=k_1*x_2+k_2*x_3+k_3*x_4$

$z_3=k_1*x_3+k_2*x_4+k_3*x_5$

$z_4=k_1*x_4+k_2*x_5+k_3*x_6$

PARTICLE-LEVEL CNN

- Particle-level CNN (P-CNN)
  - one dimensional CNN over a sequence of particles



$z_1=k_1*x_1+k_2*x_2+k_3*x_3$

$z_2=k_1*x_2+k_2*x_3+k_3*x_4$

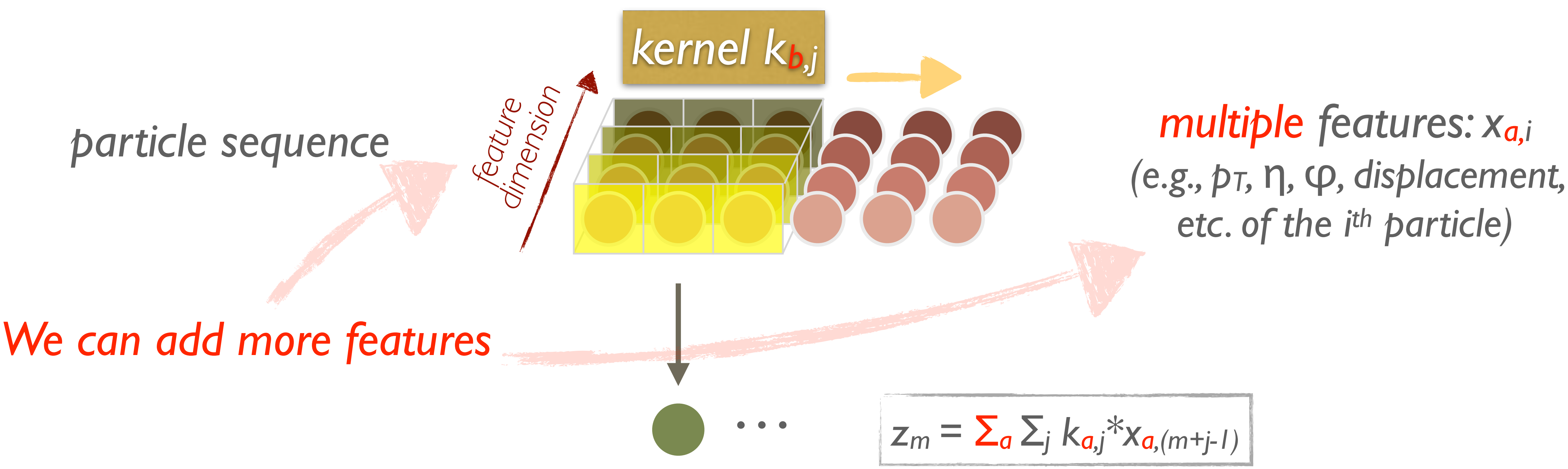
$z_3=k_1*x_3+k_2*x_4+k_3*x_5$

$z_4=k_1*x_4+k_2*x_5+k_3*x_6$

$\rightarrow$ $z_m = \Sigma_j \, k_j*x_{(m+j-1)}$

Particle-level CNN (II)

- Particle-level CNN (P-CNN)
  - one dimensional CNN over a sequence of particles



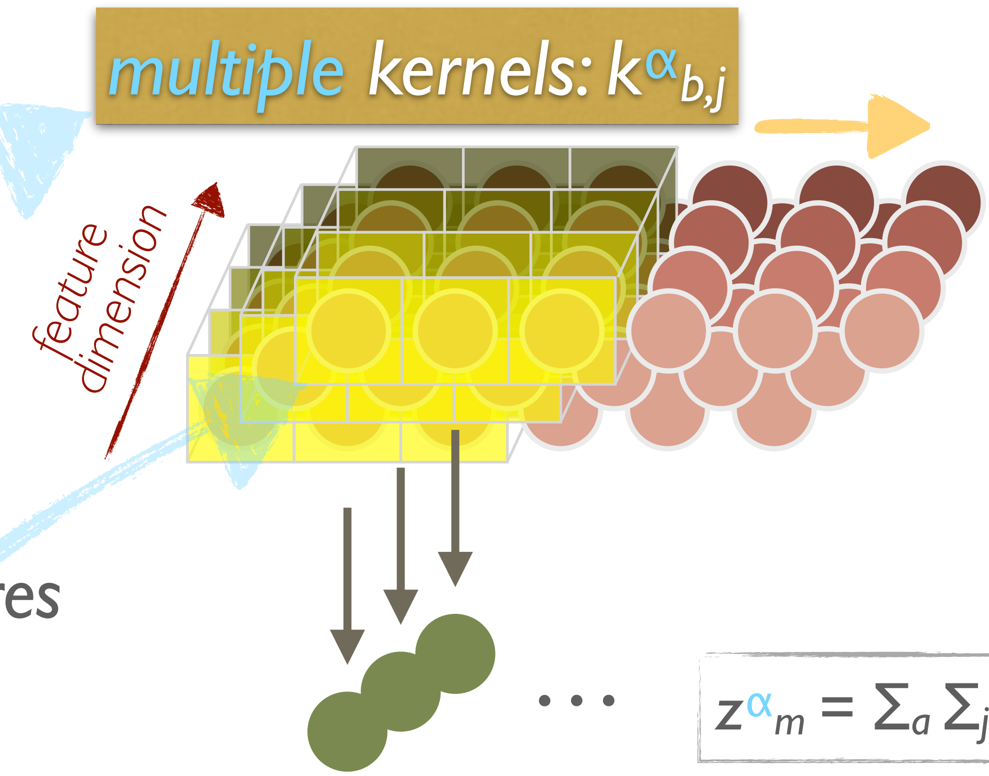
$z_m = \Sigma_a \Sigma_j k_{a,j} * x_{a,(m+j-1)}$

- *a transformation that combines all features of nearby particles*

PARTICLE-LEVEL CNN (III)

- Particle-level CNN (P-CNN)
  - one dimensional CNN over a sequence of particles

particle sequence



multiple features: $x_{a,i}$
(e.g., p_T, η, φ, displacement, etc. of the i^{th} particle)

We can add more features

. . .

$z^{\alpha}_{m} = \Sigma_a \Sigma_j k^{\alpha}_{a,j} * x_{a,(m+j-1)}$

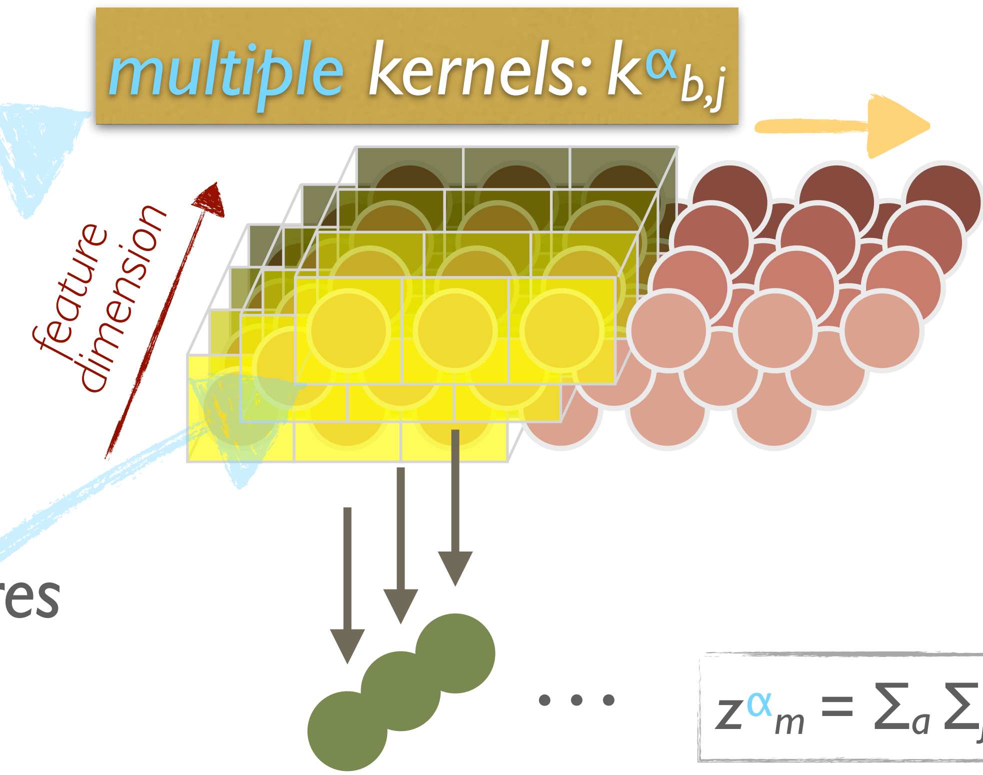
and more kernels

- *each kernel gives a different transformation*

PARTICLE-LEVEL CNN (III)

- Particle-level CNN (P-CNN)
  - one dimensional CNN over a sequence of particles

particle sequence



multiple features: $x_{a,i}$ (e.g., p_T, η, φ, displacement, etc. of the i^{th} particle)

We can add more features

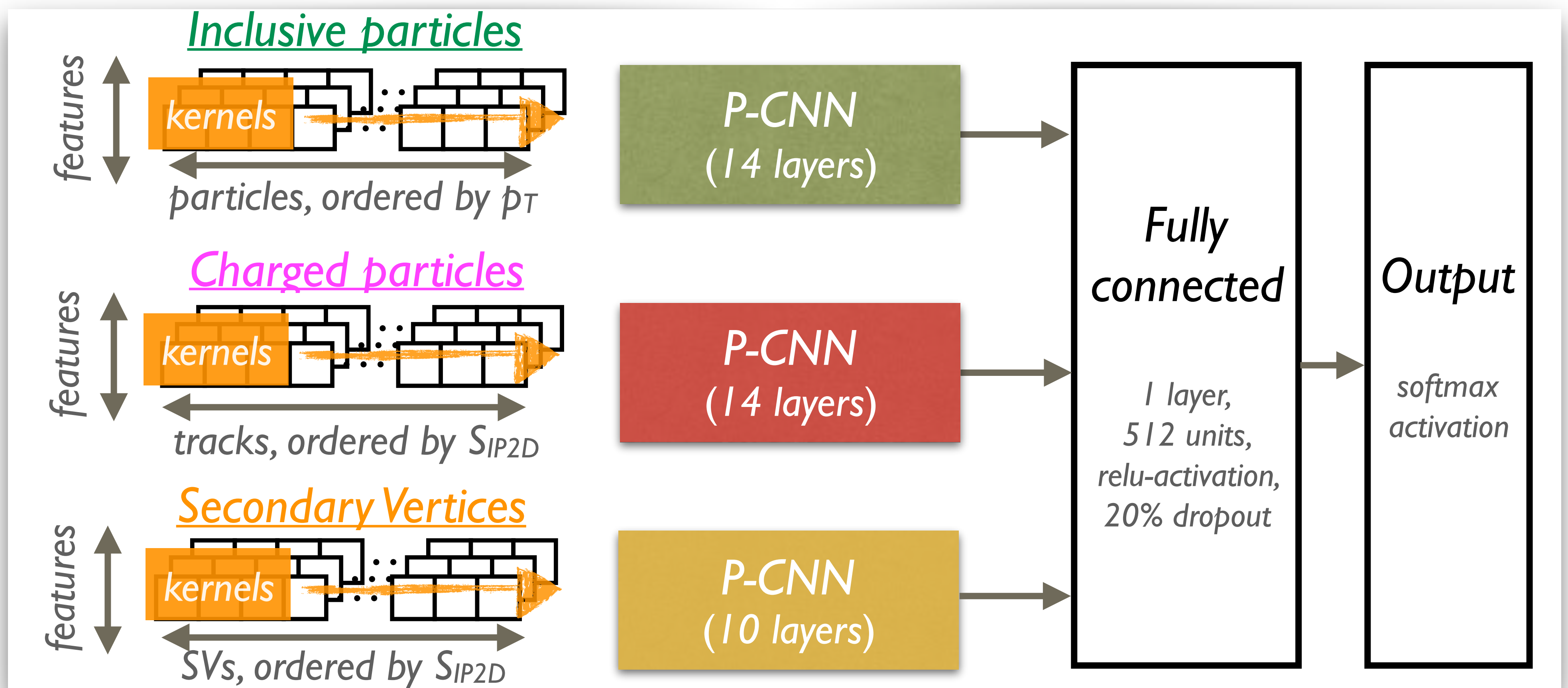
and more kernels

$$z^{\alpha}_{m} = \Sigma_a \Sigma_j \, k^{\alpha}_{a,j} * x_{a,(m+j-1)}$$

- *each kernel gives a different transformation*
- *these transformations are "learned" from the training sample to best extract information that are useful for boosted jet tagging*

ARCHITECTURE

- Inclusive particles, charged particles and SVs are first processed separately by three P-CNNs, then combined with a fully connected network to produce the final prediction



- fairly deep P-CNNs to better get the correlation between particles
- CNN architecture inspired by the ResNet model for image recognition
  - eases the training of very deep networks and improves the performance

TECHNICAL DETAILS

- Jets are clustered with anti-k_T algorithm with a distance parameter R=0.8
  - requires p_T > 400 GeV for top-tagging

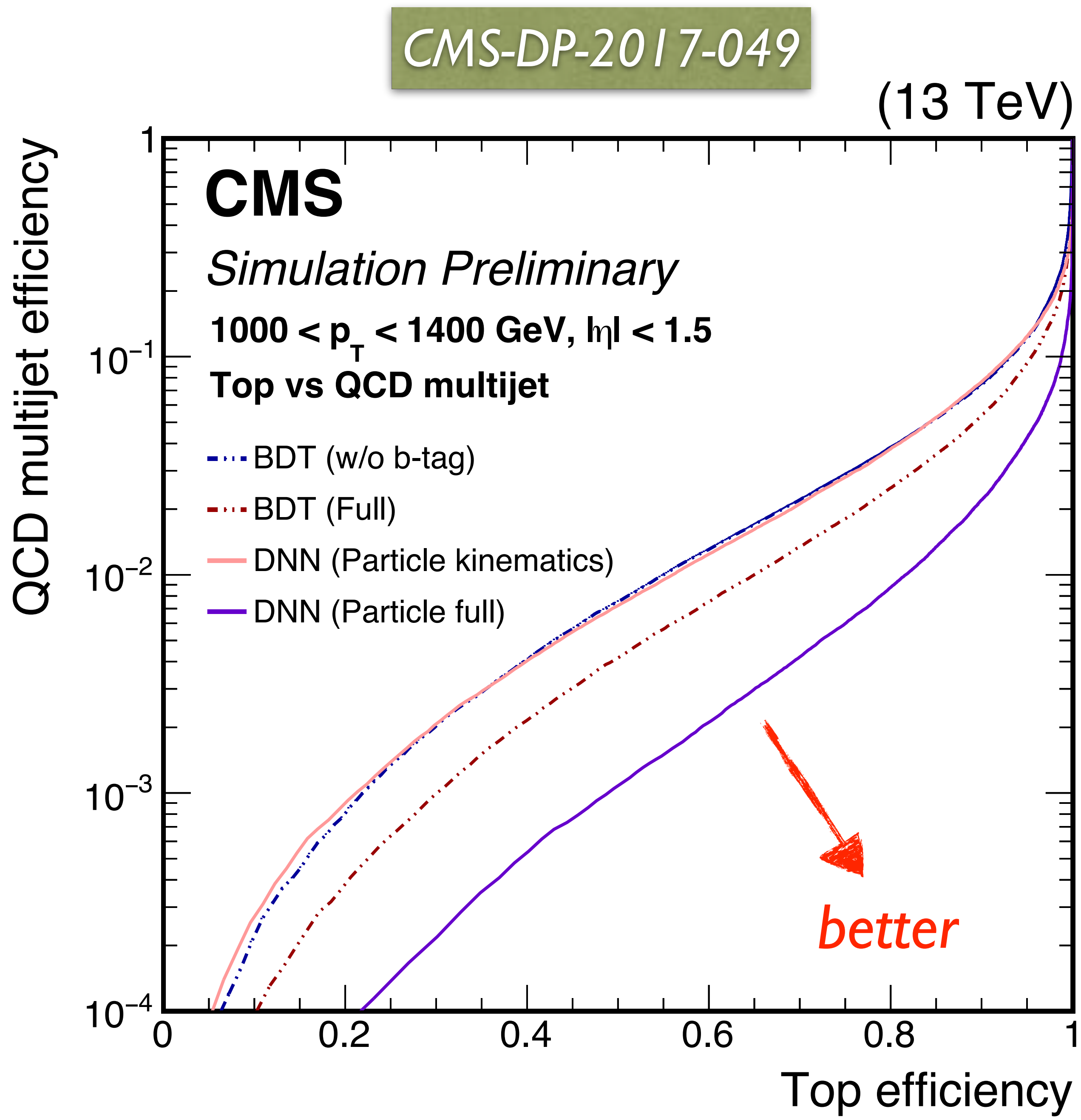
- Uses ~40 million jets from simulated samples for training
  - top quark jets taken from Z' -> tt sample
  - light quark/gluon jets taken from QCD multijet sample
  - "reweight" events to have a flat distribution in p_T to avoid bias

- Utilizes Apache MXNet package for training
  - cross entropy loss function
  - ADAM optimizer, learning rate = 0.001

PERFORMANCE

- Performance of the new DNN-based tagger is compared with alternative methods based on boosted decision trees (BDT) and jet-level observables
  - the BDT algorithm is based on the top tagger used in the CMS search for supersymmetric top quarks in the all-jets final state [CMS-SUS-16-049: JHEP 10 (2017) 005]
  - two versions with different inputs are considered
    - BDT (w/o b-tag): uses only substructure-related inputs, no “flavor” information
      - the soft drop mass, N-subjettiness ratios, observables related to quark gluon discrimination for each subjet, the relative difference in p_T between the subjets, and the mass of each subjet
    - BDT (Full): adds additional inputs related to flavor-tagging
      - e.g., the b tagging discriminator values of the subjets
- As a comparison, a simplified version of the DNN-based tagger without any flavor information is also investigated
  - DNN (Particle kinematics): uses only 6 basic kinematics variables [p_T, η, φ, ΔR(jet), ΔR(subjet1), ΔR(subjet2)] for each particle

PERFORMANCE (II)



- Flavor information helps a lot for boosted jet tagging, especially for top-tagging
  - clear improvement from BDT (w/o b-tag) to BDT (Full)
  - even larger gain from DNN (Particle kinematics) to DNN (Particle full)
- The new DNN algorithm [DNN (Particle full)] based on P-CNNs shows great power in exploiting the full set of information from jet constituents and significantly improves the performance
  - ~4x reduction in QCD multijet misidentification rate for the same top efficiency (@60%)

SUMMARY

- Presented a new method for boosted jet tagging using particle-flow jets in CMS and deep learning
  - direct use of jet constituent particles reconstructed by the particle-flow algorithm
    - try to get as much information as possible for a jet and target substructure-tagging and flavor-tagging in one go for boosted jet identification
  - a customized deep architecture based heavily on particle-level convolutional neural networks to exploit the large amount of inputs efficiently
  - significant improvement in top-tagging performance compared to alternative methods based on boosted decision trees and jet-level observables

BACKUPS

EXTRA: RESIDUAL NETWORK

- In principle, the performance of neural networks should not get worse when increasing the depth
  - deeper networks have large capacity
  - if the shallow network already achieves the best performance, the additional layers could be just identity mapping
- However, in practice, the performance of neural networks tend to degrade when it becomes very deep
- Residual network overcomes this problem by adding identity mapping between layers
  - then, the network only needs to learn the "residual" function (w.r.t the identity mapping)
- Residual network achieves the best performance at ILSVRC 2015 contest and has been widely adopted in deep neural network models
- Rerefences:
  - Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun:
    - *Deep Residual Learning for Image Recognition*, arXiv:1512.03385
    - *Identity Mappings in Deep Residual Networks*, arXiv:1603.05027