Aufbau des Codes

Aufgabe: Auf der linken Seite siehst du den schematischen Aufbau des Codes, der nachher ein *Convolutional Neural Network* zur Auswertung von OPAL-Daten trainiert. Auf der rechten Seite siehst du für einige Elemente die zugehörigen Bausteine des Jupyter Notebooks. **Verbinde** die Code-Bausteine mit den entsprechenden Bestandteilen des Codes. Wenn du fertig bist, **vergleiche** deine Lösung mit der Musterlösung.
Hinweis: Nutze gerne die Übersicht über die verschiedenen Klassen und Funktionen auf der zweiten Seite.
Achtung: Nicht für alles auf der linken Seite ist auf der rechten Seite auch ein Code-Block zu finden.

Pakete und Funktionen laden	`modell.show_learning_curve()`
Events laden	`modell.train(count_epochs= )`
Übersicht zur Analyse des Verzweigungsverhältnisses anlegen und die Rohdaten einfügen	`eventliste = load_events()`
Bilddateien in zwei Datensätze aufteilen	`eventliste_vorhersage = modell.predict(eventliste_test)`
Data Augmentation durchführen	`show_confusion_matrix(eventliste_ test, eventliste_vorhersage)`
Daten für Training und Validierung aufteilen	`overview.add_entry("Vorh", eventliste_vorhersage) overview.show()`
Übersicht erweitern	`eventliste_tv, eventliste_test = split_events_random(eventliste, fraction_first_block=)`
Modell für das Training des CNN erstellen	`overview = Overview() overview.add_entry("Roh",eventliste) overview.show()`
Training des CNN über mehrere Epochen	`faktor = eventliste_tv_vermehrt = augment_events(eventliste_tv, [faktor, faktor, faktor, faktor])`
Lernkurve des Modells anzeigen	`modell = MLModel() modell.load_structure_default() modell.show_structure() modell.load_training_eventlist( eventliste_training) modell.load_validation_eventlist(eve ntliste_validierung)`
Kategorien der Testdaten vorhersagen	
Confusion Matrix anzeigen	
Übersicht über die Verzweigungsverhältnisse um Vorhersage ergänzen	
Ereignisse, die falsch zugeordnet wurden betrachten	

Aufbau des Codes – Klassen und Funktionen

Klasse Event

Event ist eine Klasse zum speichern der Bilddaten. Ein *event* hat *filename, image* und *category.*

- *filename* ist der Dateiname, also z.B. “z5293_15219.png”
- *image* beinhaltet die eigentlichen Bilddaten als Matrix
- *category* speichert den Namen der Kategorie:
    - „q“ für Zerfälle in Quarks
    - „e“ für Zerfälle in Elektronen
    - „m“ für Zerfälle in Myonen
    - „t“ für Zerfälle in Taus

Klasse Overview

Die Klasse *Overview* ermöglicht es Übersichtstabellen zum beobachtbaren Verzweigungsverhältnis zu erstellen.

- Mit `.add_entry("Überschrift", Daten)` wird ein Eintrag zur Übersicht hinzugefügt, dabei ist der erste Parameter die Spaltenüberschrift und der zweite Parameter umfasst die Liste mit Daten.
- Mit `.show()` wird der aktuelle Zustand der verschiedenen Ereignislisten ausgegeben.
- Die Funktion `split_events_random(Daten, fraction_first_block=)` trennt eine Liste zufällig in zwei einzelne Listen. Dabei umfasst der erste Parameter die Daten und der zweite Parameter gibt an, welcher Anteil an Daten in der ersten Liste sein soll. Der Anteil wird als Dezimalzahl zwischen 0 und 1 angegeben.
- Die Funktion `.augment_events(Daten, [Faktor-Quarks, Faktor-Elektron, Faktor-Myon, Faktor-Tau])` übernimmt die Data Augmentation. Der erste Parameter umfasst die Liste an Daten und der zweite Parameter umfasst eine Liste mit den Multiplikationsfaktoren für die verschiedenen Ereignisse.
- Mit `.show_confusion_matrix(Daten, Vorhersage)` wird eine Confusion Matrix ausgegeben, bei der auf der x-Achse die vorhergesagte Kategorie und auf der y-Achse, die “echte” Kategorie stehen.
- Mit `.show_false_predictions(Daten, Vorhersage, Anzahl)` werden die Event-Displays für falsche Vorhersagen ausgegeben. Der erste Parameter umfasst die Liste mit echten Daten, der zweite die Liste mit vorhergesagten Daten und der dritte gibt an für wie viele falsche Zuordnungen das Bild ausgegeben werden soll.

Klasse MLModel

Die Klasse *MLModel* ermöglicht es ein Modell für das Training des CNN zu erstellen.

- Mit `.load_structure_default()` wird die Standard-Struktur des Modells geladen.
- Mit `.show_structure()` wird die Struktur des Modells angezeigt.
- Mit `.load_trainig_eventlist(Trainingsdaten)` werden die Trainigsdaten in das Modell geladen.
- Mit `.load_validation_eventlist(Validierungsdaten)` werden die Trainigsdaten in das Modell geladen.
- Mit `.train(count_epochs= )` wird das Modell trainiert. Der Parameter gibt dabei an, wie oft über alle Daten gegangen wird.
- Mit `.show_learning_curve()` wird die Lernkurve aufgeteilt nach Training und Validation angezeigt.
- Mit `.predict()` wird die Vorhersage der Testdaten auf Basis des Modells ausgegeben.