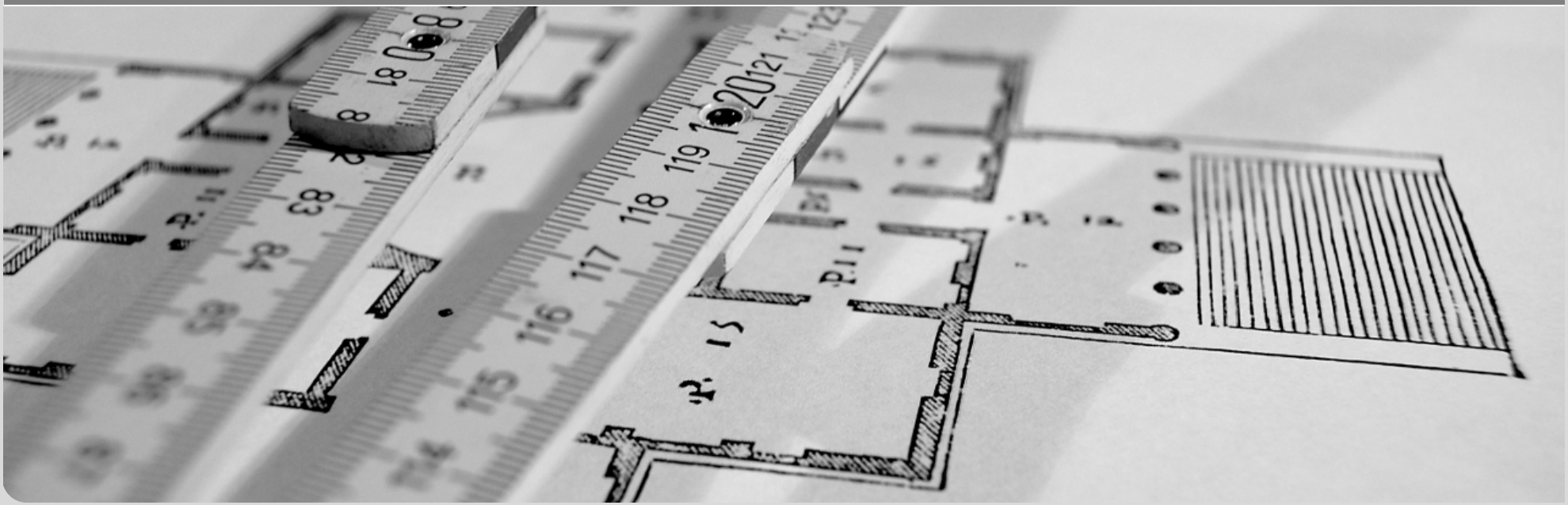
Karlsruher Institut für Technologie

Zwischenpräsentation der Java-Gruppe

Neuronale Netze mit Neuroph

Markus Braun, Daniel Hammann, Dominik Messinger, Dominic Rausch | 17. Januar 2013

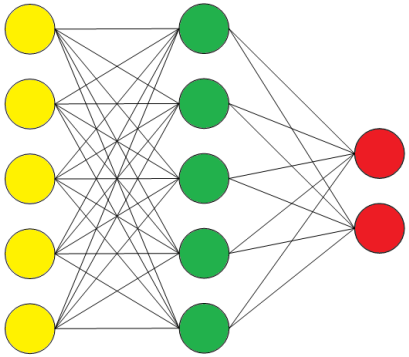
INSTITUT FÜR PROGRAMMSTRUKTUREN UND DATENORGANISATION (IPD), LEHRSTUHL FÜR PROGRAMMIERSYSTEME

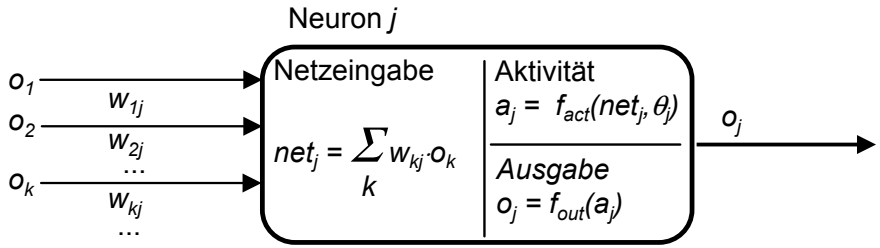


Neuronale Netze

Einleitung

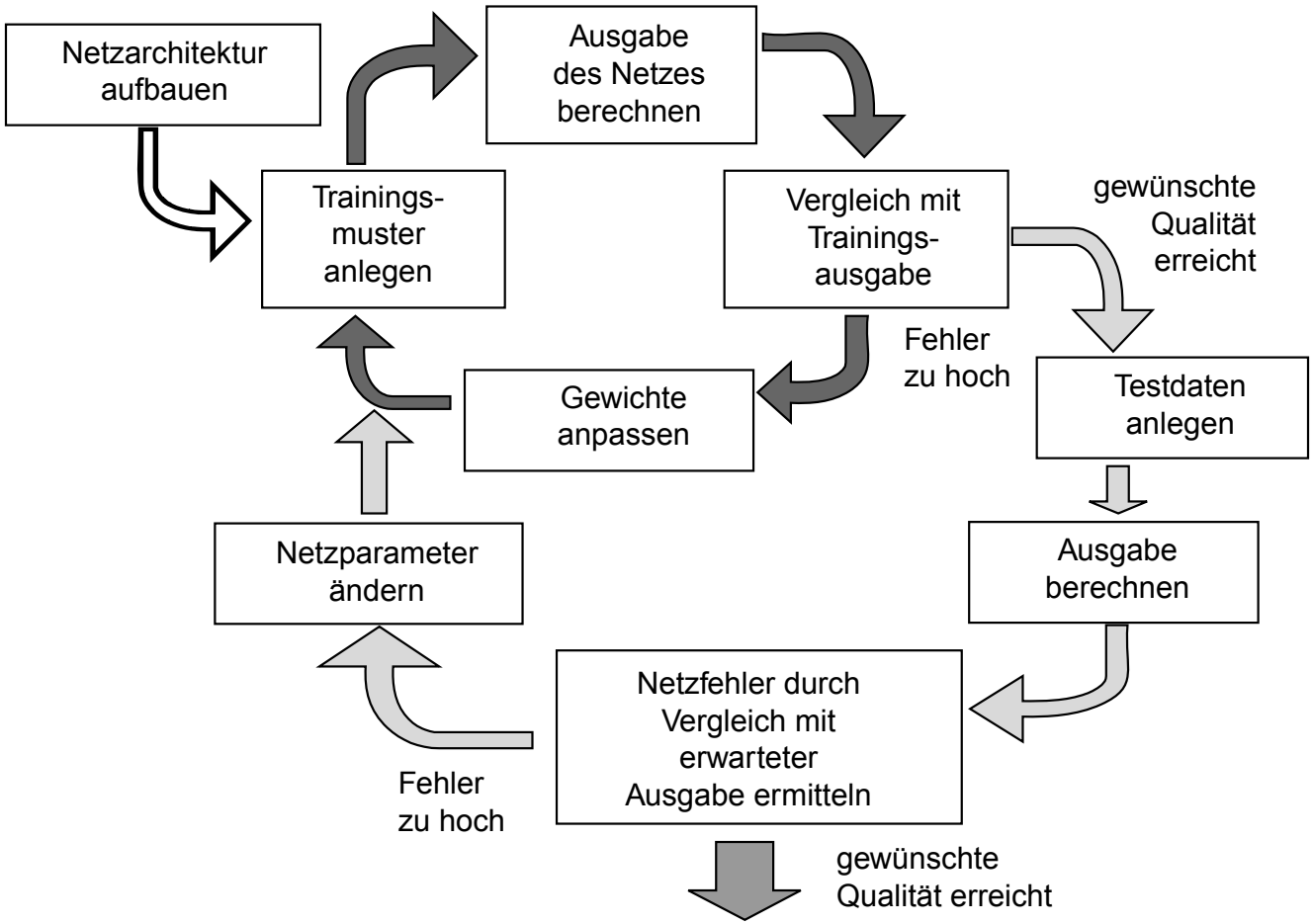
- Bestehen aus Neuronen (Verarbeitungseinheiten)
- Gewichtete Eingabeverbindungen werden zusammenfasst (Propagierungsfunktion)
- Aktivierungsfunktion mit Schwellwert
- Aus Aktivierung folgt mittels Ausgabefunktion die Ausgabe





[Quelle: Lehr- und Übungsbuch künstliche Intelligenz; Lämmel, Cleve; 2012]

Neuronale Netze



[Quelle: Lehr- und Übungsbuch künstliche Intelligenz; Lämmel, Cleve; 2012]

Überblick

Was bisher geschah

- Einarbeitung in die Thematik der Neuronale Netze
- Testdaten beschaffen
- Code Analyse der Simulationsumgebung Neuroph
- Drei Parallelisierungsversuche
- Profiling (u.a. eigenes Evaluierungsframework)

Testdaten

Erste Versuche

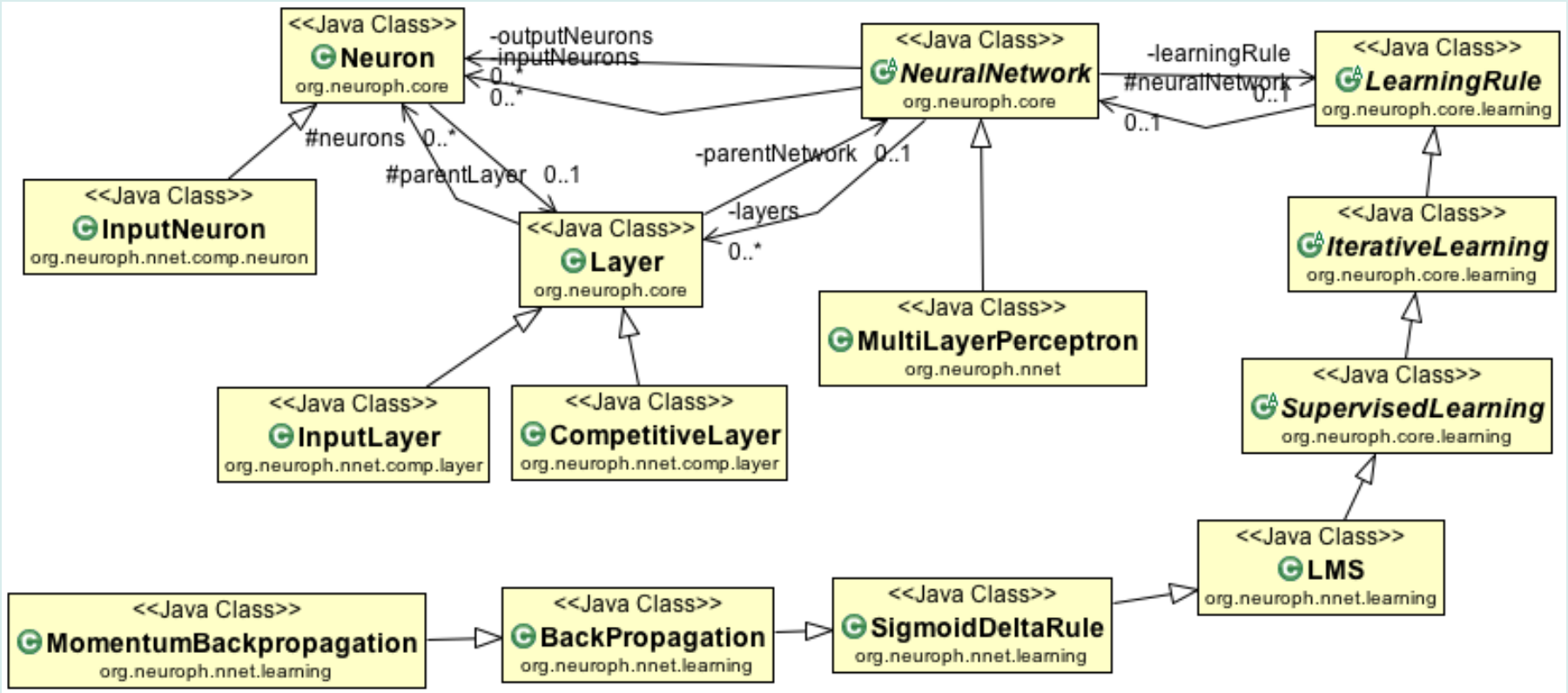
- StockExchange - Börsenvorhersage
- IrisScan Datensatz

Teilchenkollision (Cern)

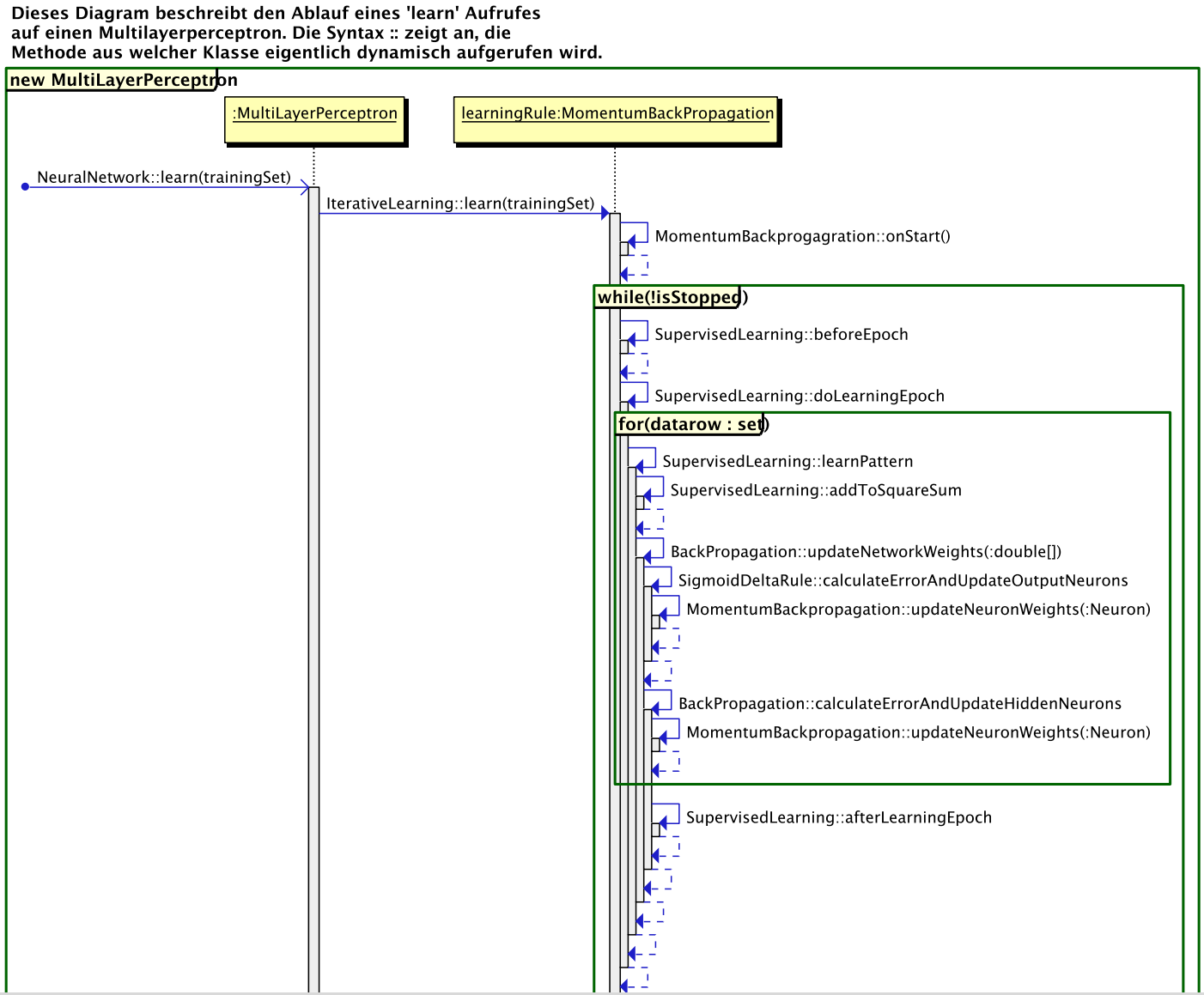
- 15k Datensätze
- Eingabe: 2853 Sensorwerte
- Ausgabe: Ist das Ereignis interessant oder nicht?

Code Analyse & Profiling

Was wir vorgefunden haben



Code Analyse: Sequenzdiagramm



Code Analyse & Profiling

Profiling

Methoden mit höchster akkumulierter Rechenzeit:

- *Connection.getInput*()
- *Connection.getWeightedInput*()

→ Zeit korreliert mit Anzahl der *Connections* und Anzahl der Lern-Iterationen

Parallelisierungsansätze

- Layer-Partitionierung
- Batch Learning Parallelisierung
- Clonebased Parallelisierung

Layer-Partitionierung

- Ansatz:
  - Parallel Auswertung innerhalb partitionierter Neuronenschichten
- Evaluation:
  - Zu feingranular, Concurrency-Overhead überwiegt erhoffte Zeitersparnis

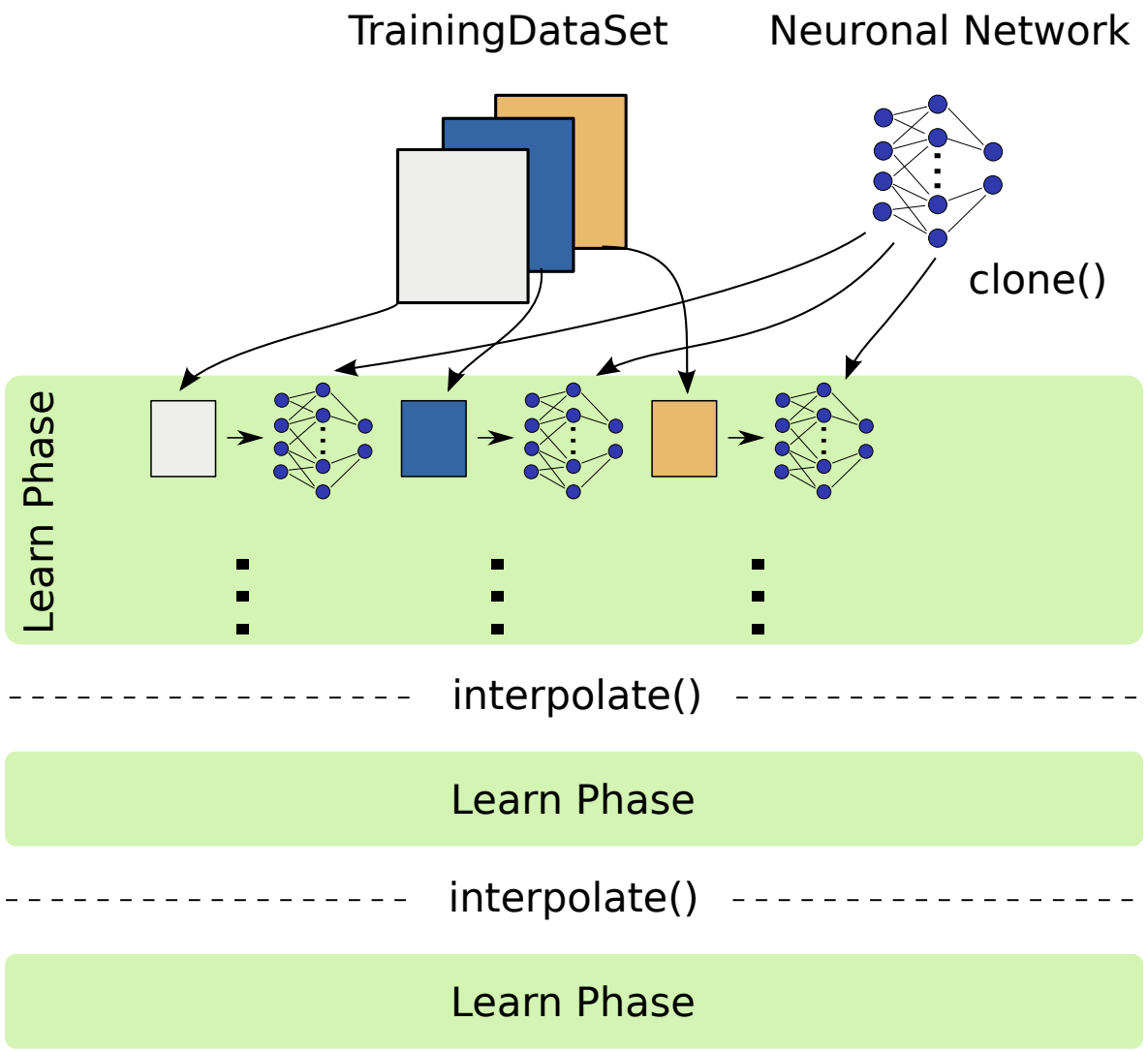
Batch Learning Parallelisierung

- Batch Learning
  - Summieren der Deltas (pro Verbindung)
  - Aktualisieren erst nach einer Epoche
- Ablauf:
  - Aufteilen der Lerndaten
  - Jeder Thread summiert Deltas
  - Aktualisieren der Gewichte
- Klone des NN notwendig

Batch Learning Parallelisierung

1. NN klonen (tiefe Kopie)
2. Daten aufteilen
3. *parallel:* Jeder Thread lernt seine Daten
4. *parallel:* Gewichte der Klone in das Hautp-NN mergen (und in die Klone)
5. *Barrieren*
6. Fehlerberechnung
7. Fehler < x ? Fertig : Goto 3

Clonebased Parallelisierung



Clonebased Parallelisierung

- ANN wird nach der Erzeugung „geklont“
- Ein Klon pro Thread
- Unterschiedliche Interpolationsfunktionen
- Weitere Parameter
  - Synchronisationsintervall
  - Maximale Anzahl von Iterationen
- Andere Ergebnisgewichte als bei sequentiellem Lernen
- Wrapper für ein neuronales Netz

Whats next?

Fahrplan

- Vollständige Evaluierung der Ansätze
- Experimentieren mit unterschiedlichen Konfigurationsparametern $\rightarrow$ Auswirkungen auf Speedup und Error

Evaluationsframework

Score

wird bestimmt durch

- Fehler (auf Testdaten)
- Laufzeit

Vorgehen

1. Permutation der Daten
2. Aufteilung in Trainings- und Testdaten
3. *foreach* ILearner L *do*
   - Lerne Trainingsdaten und messe Ausführungszeiten
   - Berechne Fehler auf Testdaten
4. Wiederhole ab 1 bis gewünschte Anzahl an Läufen erreicht

Aktuelle Evaluationswerte

Experiment-Konfiguration

- 5000 Datenreihen aus CERN-Satz, 1:1 Trainings-/Testdaten
- Intel Core2Quad Q6600, 4 Kerne à 2,4GHz, 8GB RAM
- JDK 7