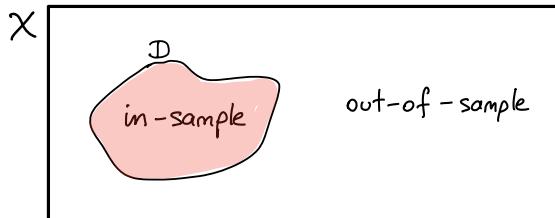


## • TRAINING UND TESTING

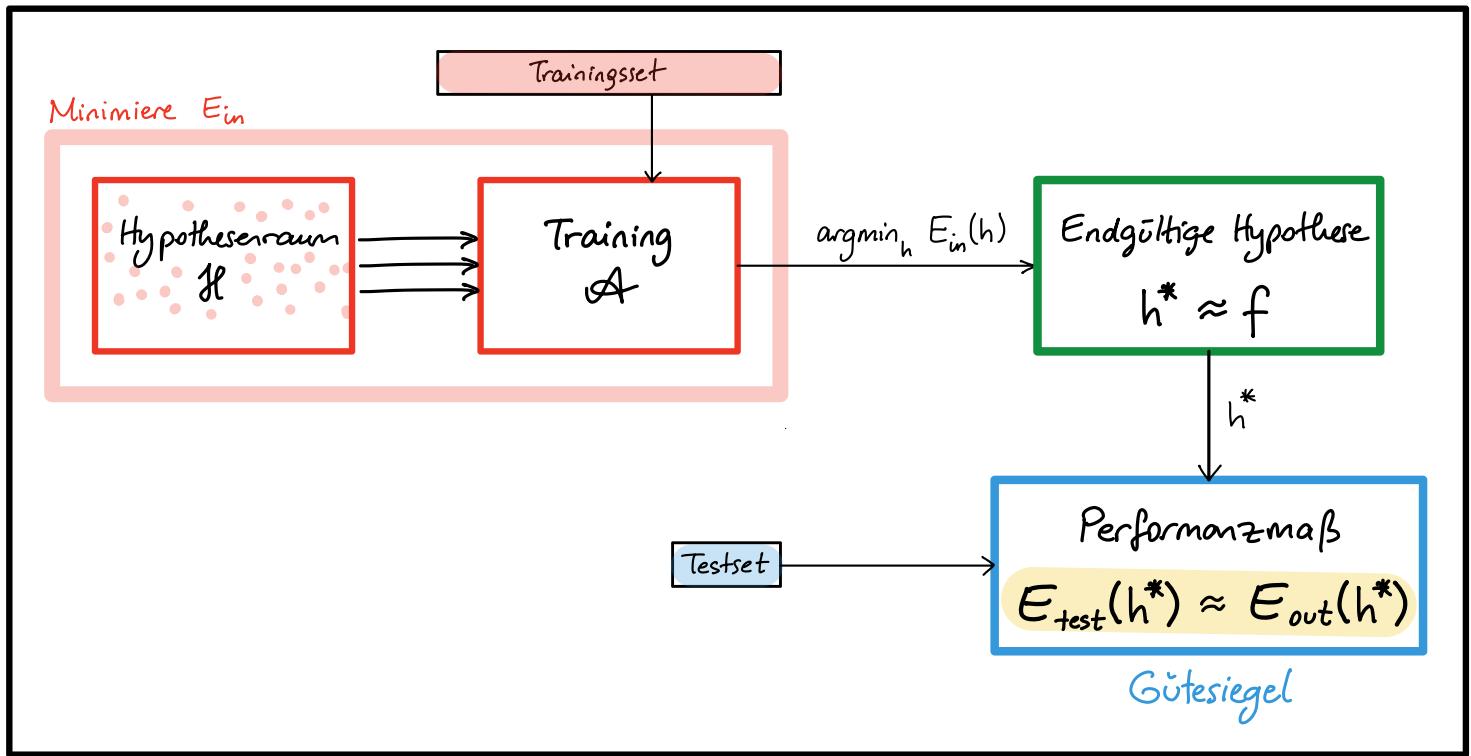
- Wann ist eine Hypothese besser als die andere? (Performanzmaß!)
- Je kleiner der Trainingsfehler, desto besser? (Overfitting!)

**ERFOLG** = Geringe Kosten auf ungesiehten Daten  
(out-of-sample error :=  $E_{\text{out}}$ )

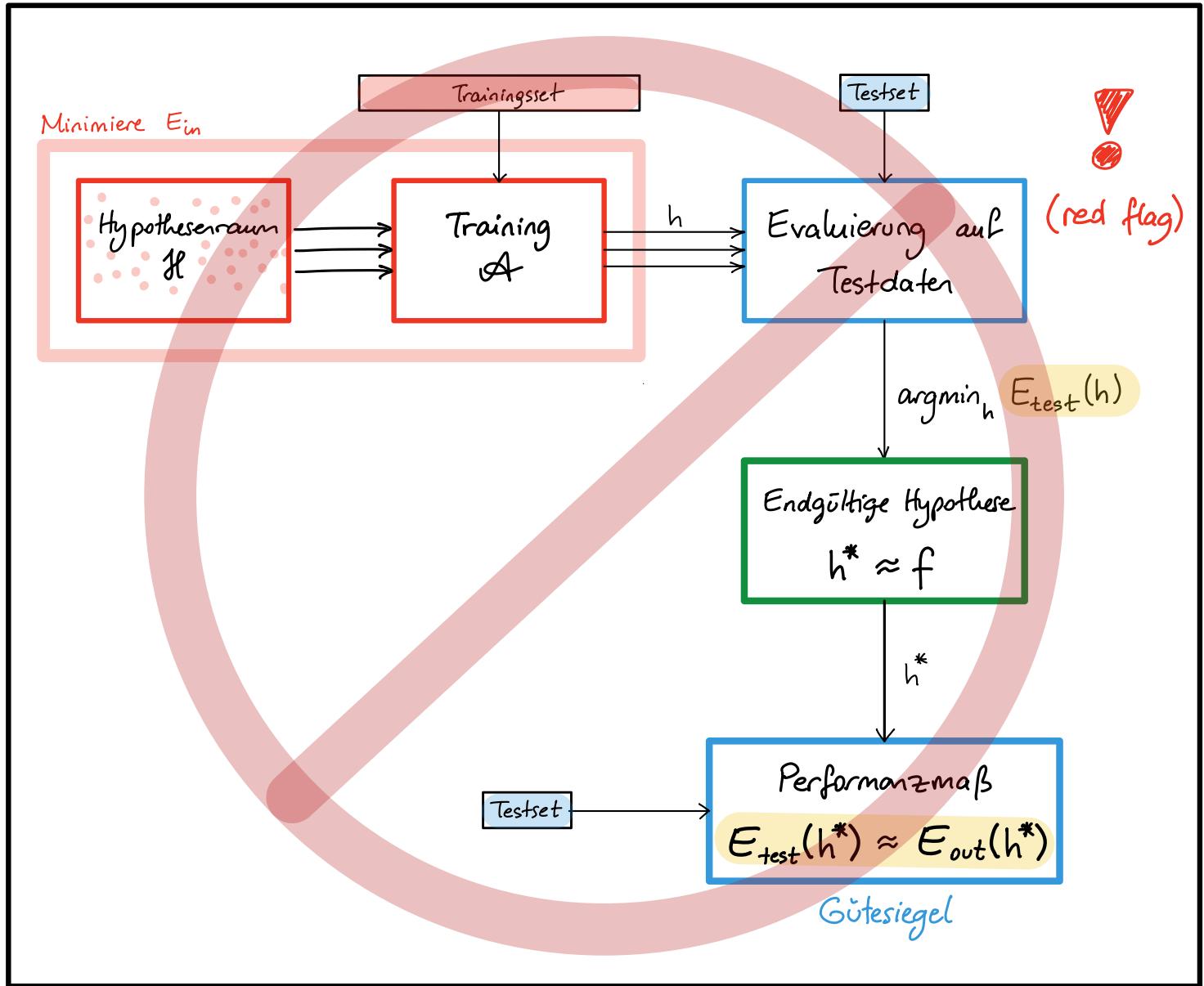


- Training = Anpassung an die Daten (Lernen!)  
= Minimierung des Trainingsfehlers (in-sample error =:  $E_{\text{in}}$ )  
Analogie: Probefüfung und Übungen
- Testing = Evaluierung der Vorhersagen für nicht geschene Daten  
= Bestimmung des Testfehlers ( $E_{\text{test}}$ )  
(einen Teil der Daten als Testdaten beiseite legen)  
Analogie: Abschlussprüfung
- Nach der "Theorie der Generalisierung" ist  $E_{\text{test}}$  eine Näherung für den "out-of-sample" Fehler  $E_{\text{out}}$  ([Abu Mostafa 2012, S.39]):

$$E_{\text{test}} \approx E_{\text{out}}$$



- Wie gut ist die Näherung  $E_{test}$ ?
  - Trainings- und Testdaten sollten möglichst aus derselben Verteilung wie die zukünftigen "real-life" Daten kommen.
  - Je mehr Testdaten, desto besser die Näherung.
  - Nach der Hoeffding Ungleichung gilt: hat man 1000 Datenpunkte im Testset, liegt  $E_{test}$  mit 98% Wahrscheinlichkeit in der 5%-Umgebung von  $E_{out}$  (ohne Beweis). [AbuMostafa 2012, S. 60]
- $E_{test} \gg E_{in}$   $\Rightarrow$  Kennzeichen für Überanpassung !  
WAS TUN ?



Achtung: Die Testdaten dürfen die Auswahl der endgültigen Hypothese in keiner Weise beeinflussen!



CHECK: Hatte man zufällig eine andere Menge von Testdaten ausgewählt, könnte sich dadurch die endgültige Hypothese  $h^*$  ändern?



## • VALIDIERUNG UND MODELLAUSWAHL

- ZIEL: Modell mit bester Generalisierung (kleinstem  $E_{out}$ ) bestimmen.

ABER:  $E_{out}$  ist unbekannt und die Näherung  $E_{test}$  darf nicht bei der Modellauswahl eingesetzt werden!

- LÖSUNG: Train / Val / Test Aufteilung (engl. split)

Einen weiteren Teil der Trainingsdaten als Validierungsset (auch development set) beiseite legen.

Den Fehler  $E_{val}$  auf diesen Validierungsdaten für die Diagnose und Vermeidung von Overfitting einsetzen!

- Der Validierungsfehler soll als Wegweiser bei Entscheidungen dienen:

- Auswahl der Hypothesenmenge  $\mathcal{H}$ 
  - # Schichten, # Zellen pro Schicht, Aktivierungsfunktionen
- Auswahl des Lernalgorithmus  $\mathcal{A}$ 
  - Kostenfunktion, Regularisierungsparameter
- Auswahl der Parameter ( $\theta$ )
  - Unter allen durchlaufenen Modellen  $h \in \mathcal{H}$ , jenes mit minimalem  $E_{val}$  auswählen

- Mögliche Aufteilungen:

- $|D| \approx 100.000$



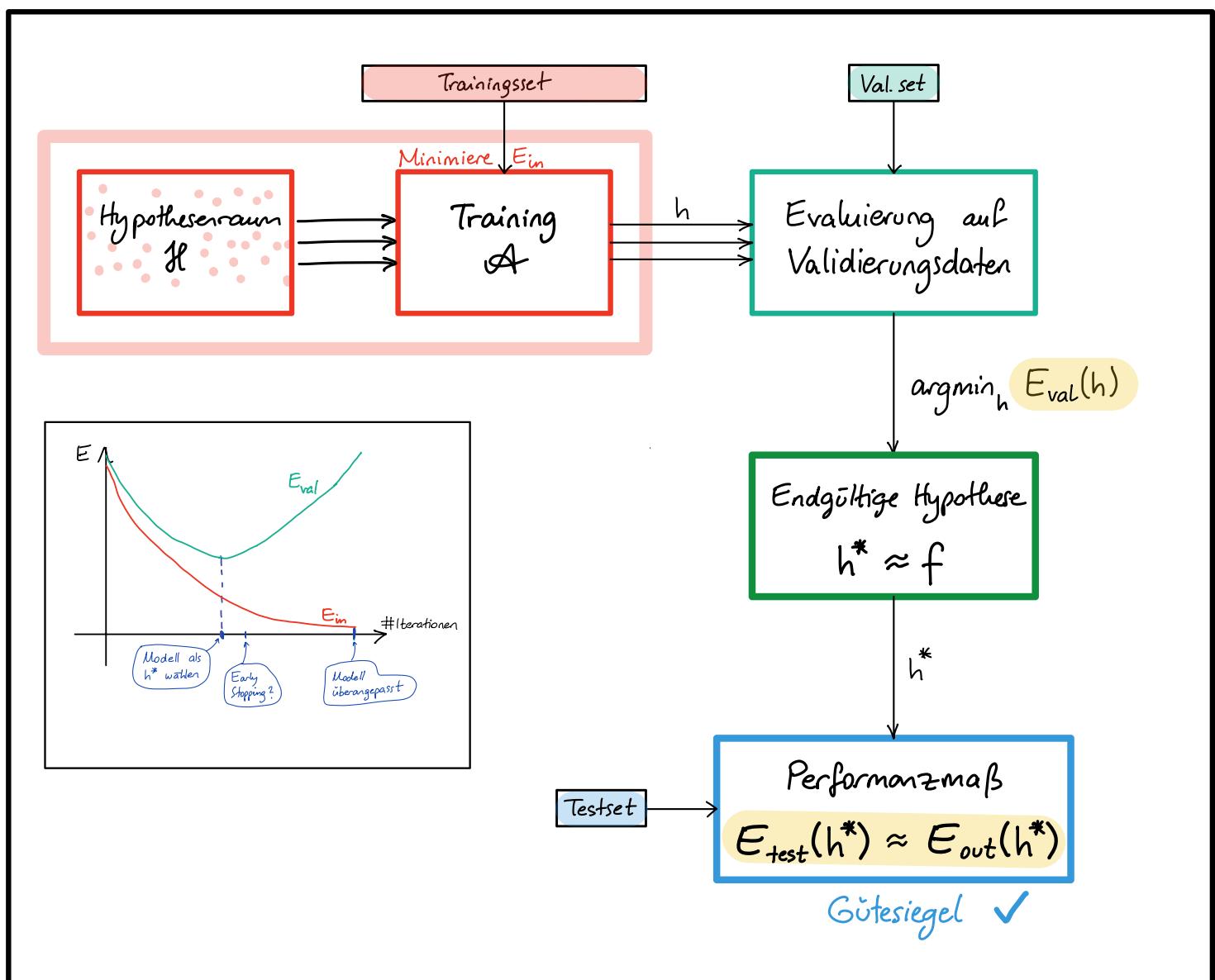
- $|D| \approx 10.000.000$



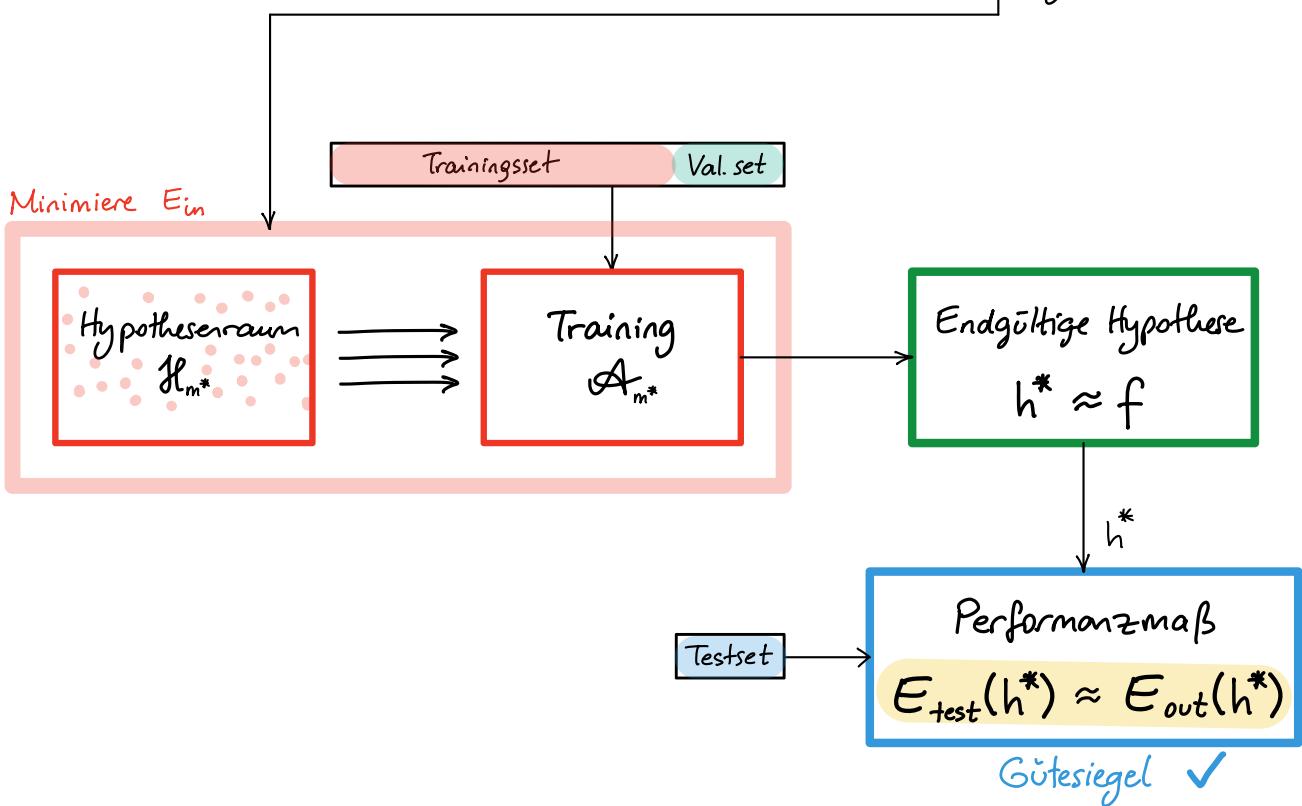
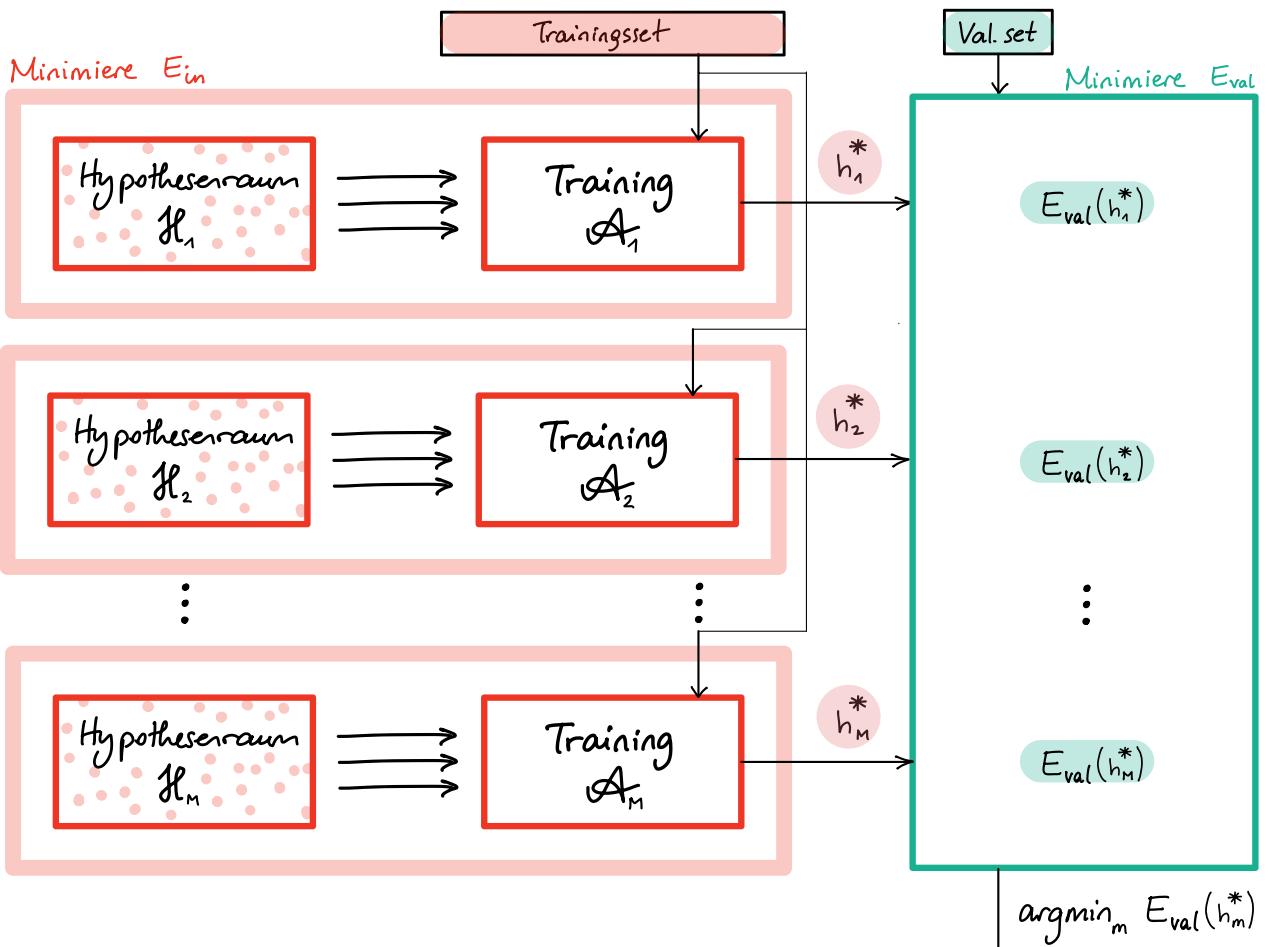
- Wenn man kein Gütesiegel braucht, kann man auf die Testdaten verzichten:



**BSP** Einsatz der Validierungsdaten bei der Parameterauswahl



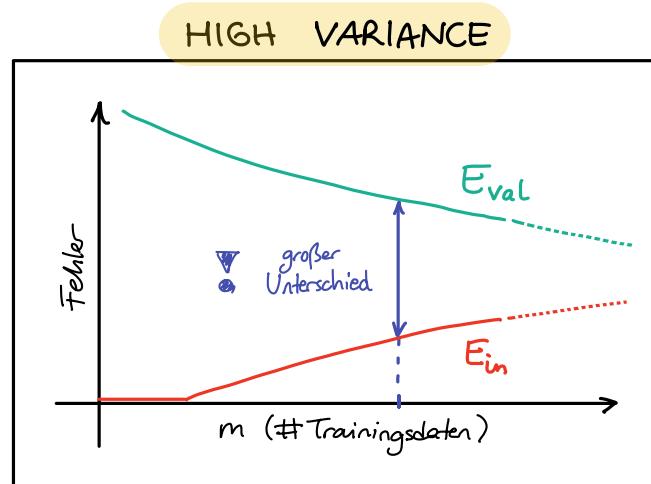
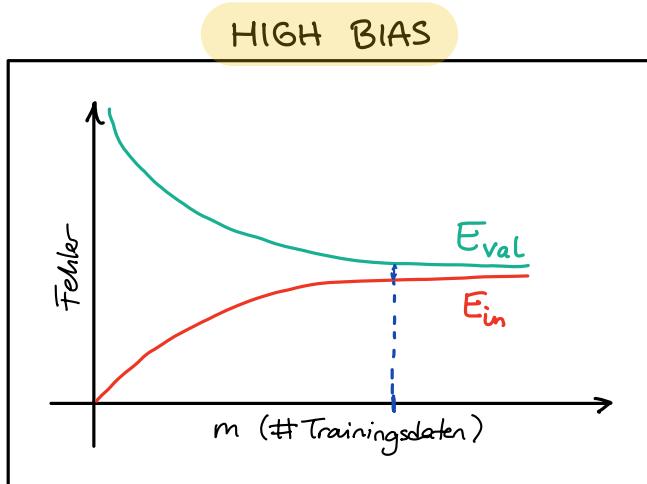
# BSP: Einsatz der Validierungssachen für das Hyperparameter-Tuning



- Verschiedene Modelle ( $\mathcal{H}_m, \omega_m$ ) trainieren,  
das Modell ( $\mathcal{H}_{m^*}, \omega_{m^*}$ ) mit kleinstem  $E_{\text{val}}$  auswählen  
und zuletzt auf (Trainingsset + Validierungsset) trainieren;  
das Ergebnis als endgültige Hypothese zurückgeben.
- Bemerkung :
  - Ist Validierungsset oder Trainingsset zu klein, führt das zu einer schlechten Hyperparameter-Auswahl.
  - Gutes Trade-off finden!  
Oder: k-fache Kreuzvalidierung verwenden.

BSP.

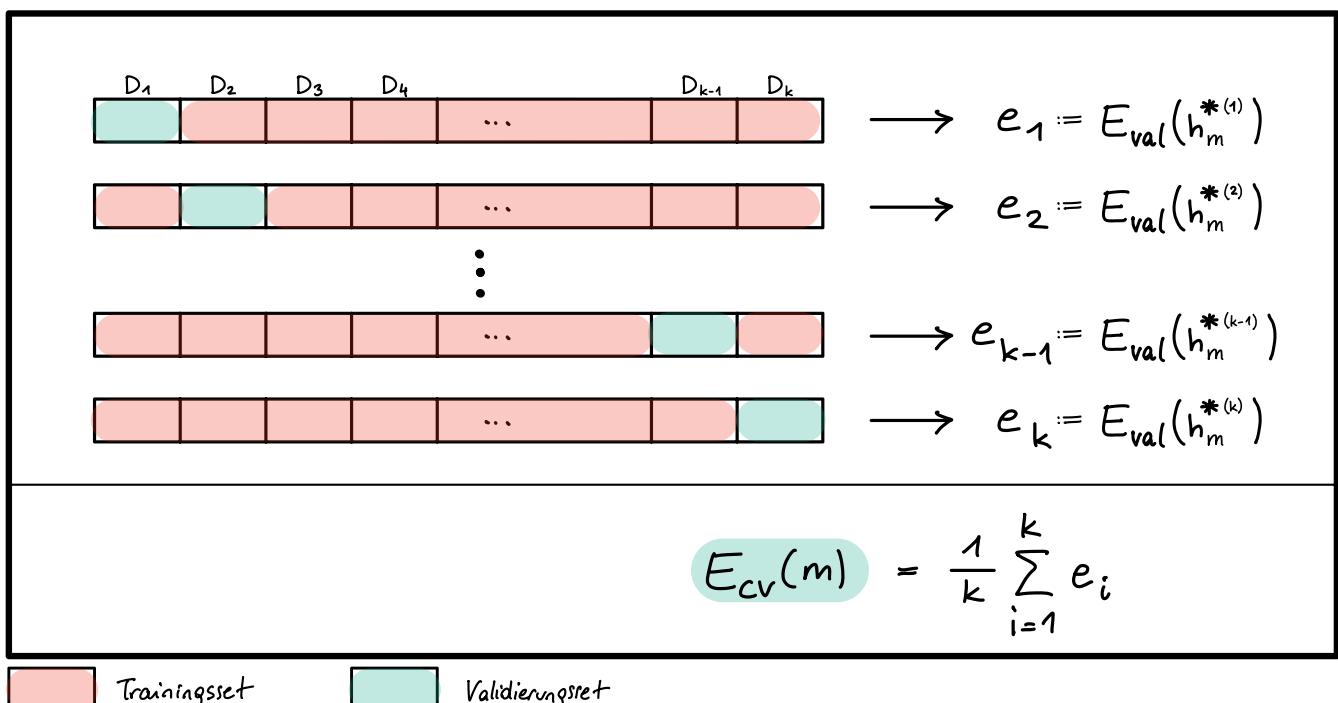
## LERNKURVEN



- Unteranpassung  
(Einfaches Modell)
- Mehr Daten helfen nicht
- Für  $m \rightarrow \infty$  :  $E_{\text{val}} = E_{\text{in}}$
- Überanpassung  
(Komplexes Modell)
- Mehr Daten könnten helfen
- Für  $m \rightarrow \infty$  :  $E_{\text{val}} = E_{\text{in}}$   
 $E_{\text{val}} \gg E_{\text{in}}$

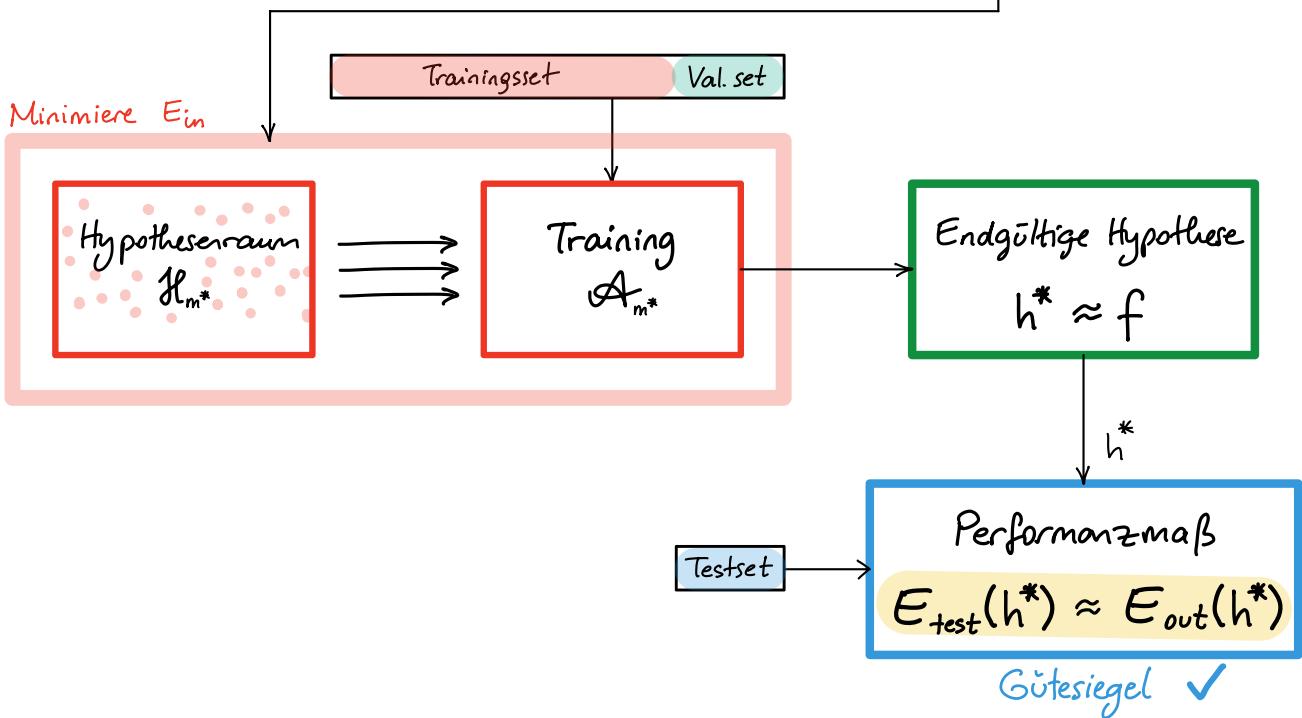
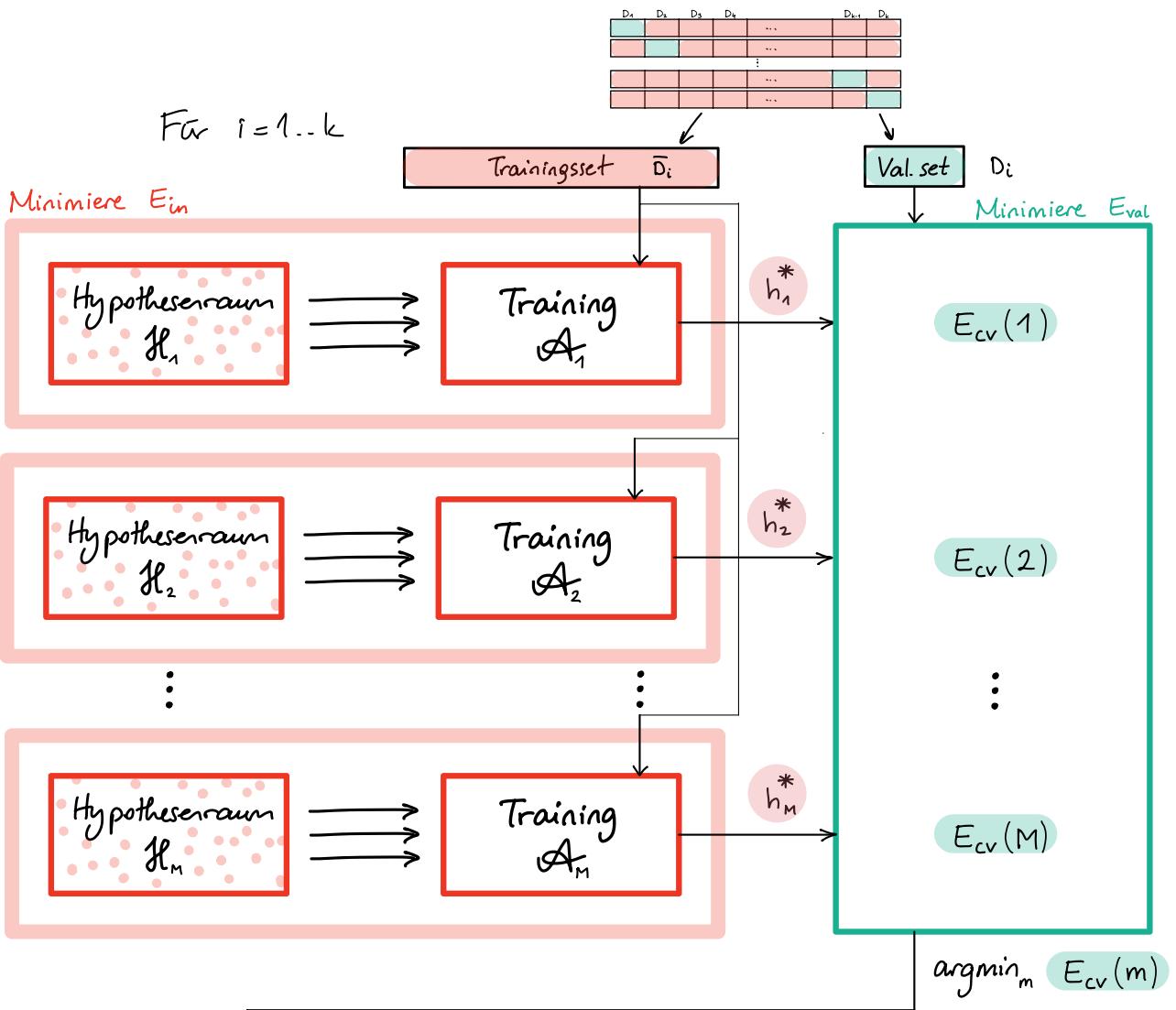
## • K-FACHE KREUZVALIDIERUNG (engl. k-fold cross-validation)

- Trainingsdaten in  $k$  gleichgroße Teilmengen  $D_1, D_2, \dots, D_k$  zerlegen.  
Für jedes  $(\mathcal{H}_m, \mathcal{A}_m)$  das Training  $k$  mal wiederholen;  
für das  $i$ -te Training die Teilmenge  $D_i$  für die Berechnung  
des Validierungsfehlers  $e_i := E_{\text{val}}(h_m^{*(i)})$ , den Rest für  
das Training benutzen.
- Den Durchschnitt der  $k$  Validierungsfehler  $e_1, \dots, e_k$  als den  
Kreuzvalidierungsfehler  $E_{\text{cv}}(m)$  des Modells  $(\mathcal{H}_m, \mathcal{A}_m)$  berechnen:



- Das  $(\mathcal{H}_{m^*}, \mathcal{A}_{m^*})$  mit  $m^* = \operatorname{argmin}_m E_{\text{cv}}(m)$  auswählen,  
zuletzt auf (Trainingsset + Validierungsset) trainieren;  
das Ergebnis als endgültige Hypothese zurückgeben.

# BSP: Einsatz der Validierungsdaten für das Hyperparameter-Tuning



- Bemerkung:  
 $E_{cv}$  liefert eine bessere Näherung (vermeidet Überanpassung an eine Validierungsmenge und "sieht" alle Daten in der Trainingsmenge), ist aber rechenintensiver.

**BSP**

$\mathcal{H}$ : Neuronales Netzwerk mit zwei verdeckten Schichten mit jeweils 100 und 50 Knoten und ReLU als Aktivierung.

$\Delta$ : Gradient Descent, Cross-Entropy Loss, Lernrate  $\alpha = 0.01$ .  
 Regularisierungsparameter  $\lambda$  soll justiert werden (fine-tune)  
 ( $\lambda$  ist ein Hyperparameter von  $\Delta$ )

$$\begin{array}{ll}
 (\mathcal{H}, \Delta_1) \text{ mit } \lambda = 10 & \left( \begin{array}{ccc} \rightarrow h_1^* & \rightarrow E_{val}(h_1^*) \\ \rightarrow h_2^* & \rightarrow E_{val}(h_2^*) \\ \rightarrow h_3^* & \rightarrow E_{val}(h_3^*) \\ \rightarrow h_4^* & \rightarrow E_{val}(h_4^*) \end{array} \right) \\
 (\mathcal{H}, \Delta_2) \text{ mit } \lambda = 1 & \left( \begin{array}{c} E_{cv}(1) \\ E_{cv}(2) \\ E_{cv}(3) \\ E_{cv}(4) \end{array} \right) \\
 (\mathcal{H}, \Delta_3) \text{ mit } \lambda = 0.1 & \\
 (\mathcal{H}, \Delta_4) \text{ mit } \lambda = 0.01 &
 \end{array}$$

$(\mathcal{H}, \Delta_2)$  mit  $\lambda = 1$   $\xrightarrow[\text{train+val set}]{\text{training mit}}$   $h^*$  endgültige Hypothese.