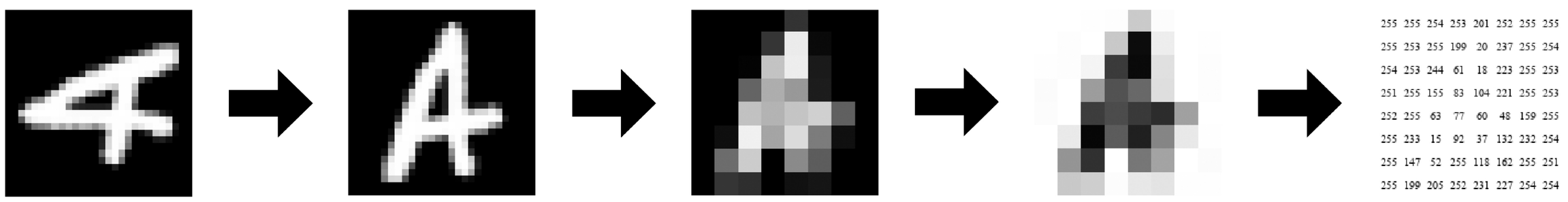
SCHRIFTERKENNUNG MIT HILFE VON NEURAL NETWORKING

EINE ARBEIT VON
ARNE VIRIDEN
BETREUT VON
YVES WEBER

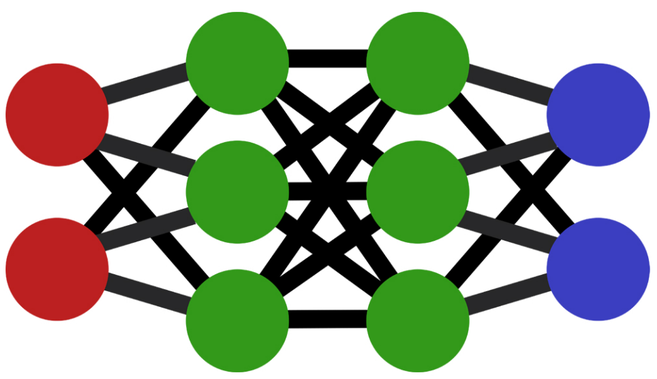
Wie lässt sich das Prinzip von Neural Networking nutzen, um Handschrift zu erkennen?

```
125 modelcache[layer]["Biases"][node] -= delta
126
127 #Backward Propagation für Weights, wenn nicht Input-Layer
128 if layer != 0:
129     for prevnode in range(self.architecture[layer - 1]):
130         #L2-Regularisierung
131         l2penalty = l2lambda * self.model[layer]["Weights"][node][prevnode] / minibatchsize
132
133         #Speichert den Neuron-Fehler des vorherigen Layers
134         errors[layer - 1][prevnode] += self.model[layer]["Weights"][node][prevnode] * nodederivative
135
136         #Backward Propagation für das Weight
137         activation = applyactivation(output[layer - 1][prevnode], False)
138         modelcache[layer]["Weights"][node][prevnode] -= (activation * delta + l2penalty)
```

Umgesetzt in Python Code (ohne externe ML-Libs)



Preprocessing der Daten

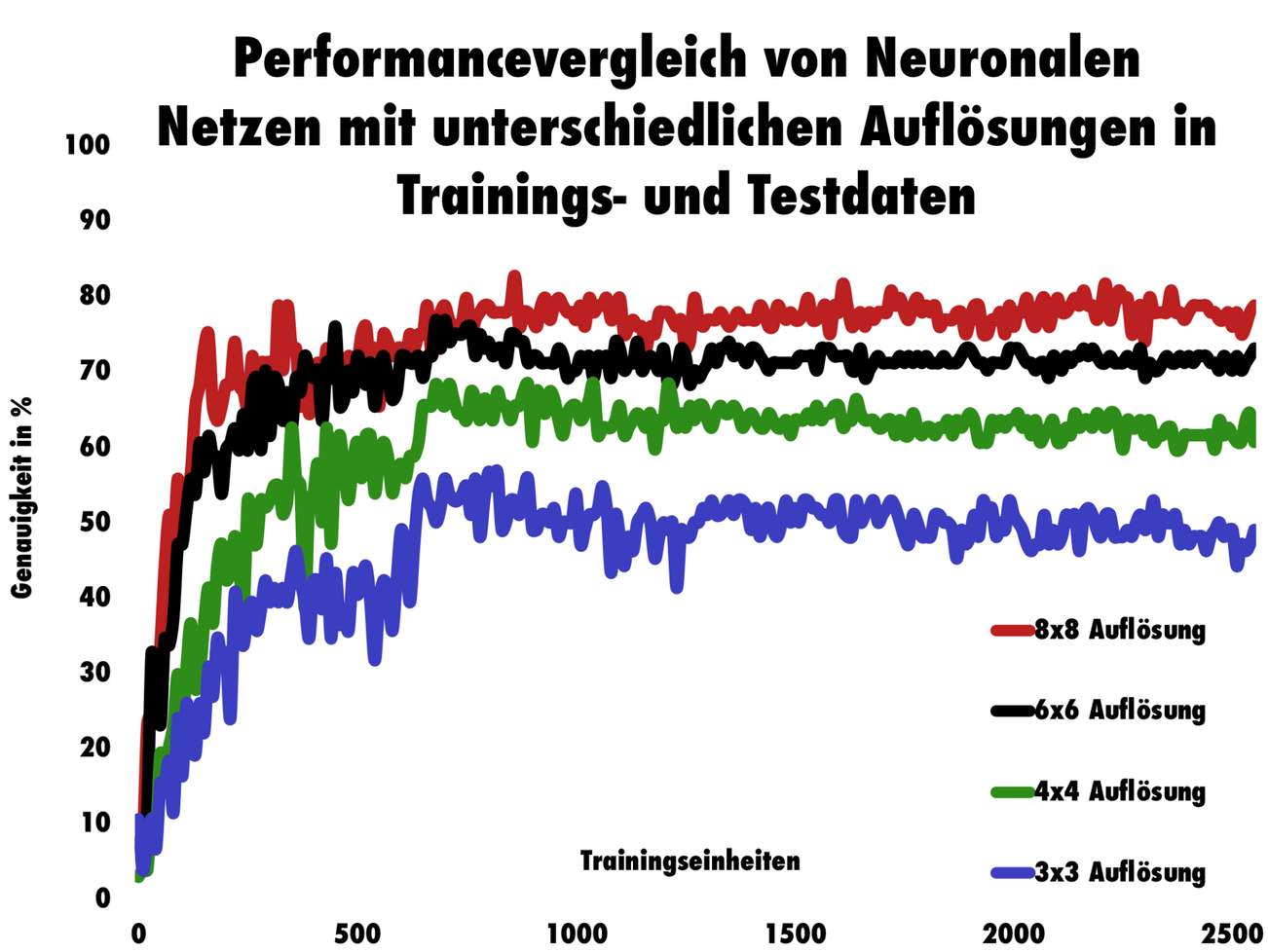


Simples Neuronales Netzwerk (=FCNN)

#n im l+1

$$\frac{\partial C}{\partial a_{l,n}} = \sum_{i} \frac{\partial z_{l+1,i}}{\partial a_{l,n}} \times \frac{\partial a_{l+1,i}}{\partial z_{l+1,i}} \times \frac{\partial C}{\partial a_{l+1,i}}$$

Backward Propagation Formel (Fehlerrechnung im Training)



Fazit: Training und Hyperparameter des Programmes sind Trade-off zw. Rechenaufwand und Leistung

Entschieden wurde sich in meiner Arbeit für ein kleineres ("dümmeres") NN, welches trotzdem 78.8% Genauigkeit erreichte

Demnach können auch simple NN mit genügendem Vorverarbeiten der Daten dezent performen

KANTONSSCHULE LIMMATTAL

Github Repo