1 Algorithmus

Input: loss function E, learning rate η, dataset X, y und das Modell $F(\theta, x)$
Output: Optimum θ which minimizes ϵ

while *converge* **do**
  $\tilde{y} = F(\theta, x)$
  $\theta = \theta - \eta . \frac{1}{N} \sum_{i=1}^{N} \frac{\delta \epsilon(y, \tilde{y})}{\delta \theta}$
end

Algorithm 1: Gradient descent

Input: loss function E, learning rate η, dataset X, y und das Modell $F(\theta, x)$
Output: Optimum θ which minimizes ϵ

while *converge* **do**
  Shuffle X, y
  for x_i, y_i *in* X, y **do**
    $\tilde{y} = F(\theta, x_i)$
    $\theta = \theta - \eta . \frac{1}{N} \sum_{i=1}^{N} \frac{\delta \epsilon(y_i, \tilde{y_i})}{\delta \theta}$
  end
end

Algorithm 2: Stochastic Gradient descent(SGD)

Input: loss function E, learning rate η, dataset X, y und das Modell $F(\theta, x)$

Output: Optimum θ which minimizes E

while *converge* **do**
  Shuffle X, y
  for *each batch of* x_i, y_i *in X, y* **do**
    $\tilde{y} = F(\theta, x_i)$
    $\theta = \theta - \eta . \frac{1}{N} \sum_{i=1}^{N} \frac{\delta E(y_i, \tilde{y_i})}{\delta E}$
  end
end

Algorithm 3: Mini-Batch Stochastic Gradient descent(MSGD)

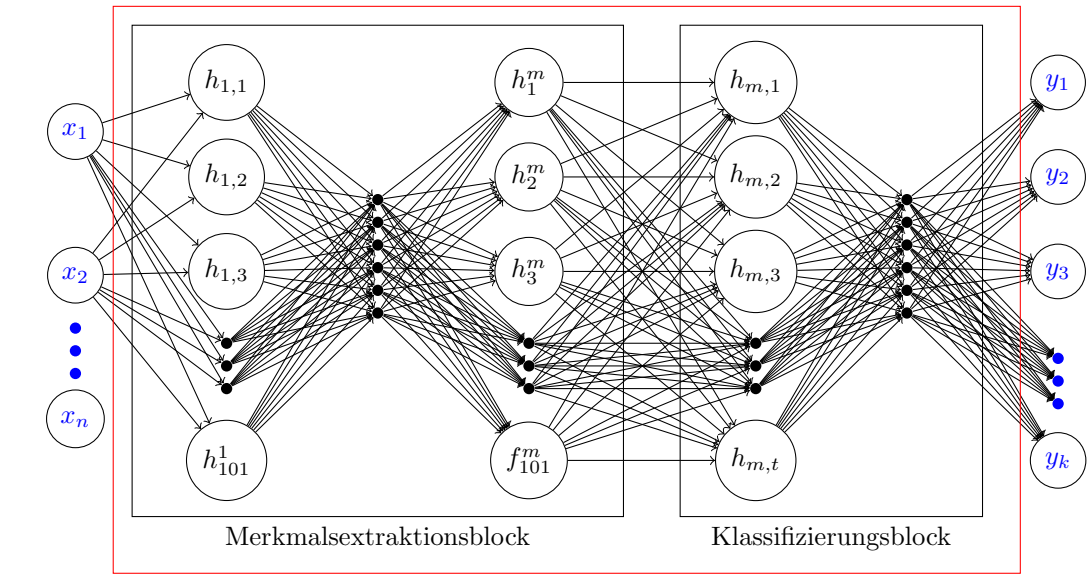
Input: Netzwerk mit l layers, Aktivirungsfunktion σ_l , Output von der verstekten Schicht $h_l = \sigma_l(W_l^T h_{l-1} + b_l)$ und die Netzwerkausgabe $\tilde{y} = h_l$

Berechnen der Gradient: $\delta \leftarrow \frac{\partial E(y_i, \tilde{y}_i)}{\partial y}$

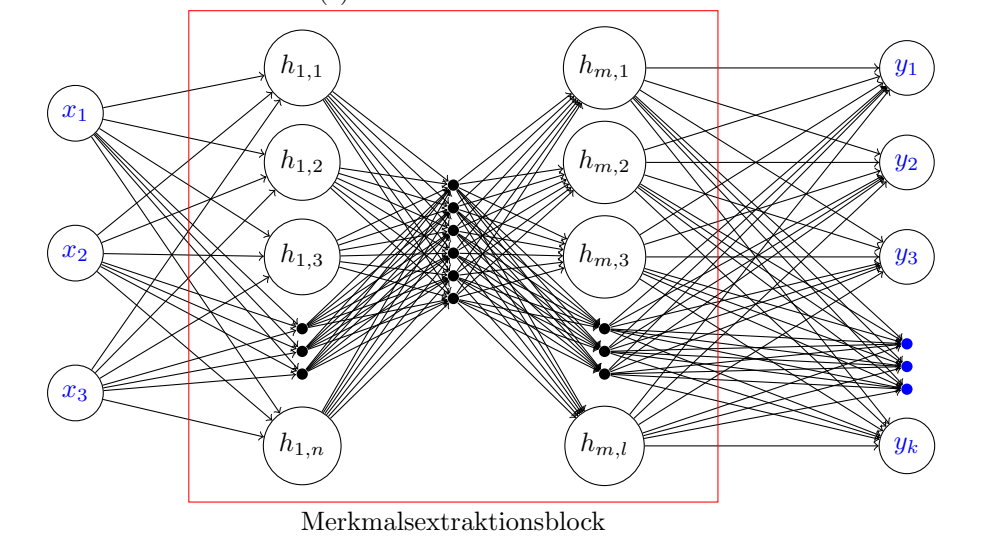
for $i \leftarrow l$ *bis* 0 **do**
  Berechnen der Gradient für die Aktuelle Schicht
  $\frac{\partial E(y,\tilde{y})}{\partial W_l} = \frac{\partial E(y,\tilde{y})}{\partial h_l} \frac{\partial h_l}{\partial W_l} = \delta \frac{\partial h_l}{\partial W_l}$
  $\frac{\partial E(y,\tilde{y})}{\partial b_l} = \frac{\partial E(y,\tilde{y})}{\partial h_l} \frac{\partial h_l}{\partial b_l} = \delta \frac{\partial h_l}{\partial b_l}$
  Gradientabstiegverfahren mit $\frac{\partial E(y,\tilde{y})}{\partial W_l}$ und $\frac{\partial E(y,\tilde{y})}{\partial b_l}$
  Propagiere den Gradienten zu den unteren Schichten.
  $\delta \leftarrow \frac{\partial E(y,\tilde{y})}{\partial h_l} \frac{\partial h_l}{\partial h_{l-1}} = \delta \frac{\partial h_l}{\partial h_{l-1}}$
end

Algorithm 4: Back-Propagation

2 Neuronale Netzwerk



(a) *TemkiNet Architektur*



(b) Allgemeine CNN-Architektur