Aufgabe 9 (Künstliches neuronales Netzwerk zur Klassifizierung handschriftlicher Ziffern)
Auf der Webseite zur Vorlesung[1] findet sich in `ann_mnist.zip` der MNIST-Datensatz und entsprechende Python-Programme zum Aufstellen und Trainieren eines künstlichen neuronalen Netzwerks.

(a) Machen Sie sich mit der Implementierung vertraut, und vervollständigen Sie den Code an den (mit `#### ...`) angegebenen Stellen in `layers.py`. Testen Sie den Code durch Aufruf von `test_layers.py`. [2 Punkte]

(b) Lassen Sie nun `run_training.py` zum Trainieren des Netzwerks anhand des MNIST-Datensatzes laufen. Wie verändert sich der Klassifizierungserfolg, wenn Sie die Anzahl der Zellen in der versteckten Schicht von 30 auf 5 reduzieren, oder (durch Modifikation von `network.py`) eine zusätzliche versteckte Schicht (etwa mit 15 Zellen) hinzufügen? [2 Punkte]

Hinweis: Die Ausgabe beim Training sollte in etwa so aussehen:

```
(Iteration 1 / 15625) loss: 1.247887
(Epoch 0 / 5) train acc: 0.105000; val_acc: 0.089200
(Iteration 1001 / 15625) loss: 0.052469
(Iteration 2001 / 15625) loss: 0.034348
...
(Epoch 5 / 5) train acc: 0.962000; val_acc: 0.955300
```

D.h. nach der letzten "Epoche" (Durchlauf durch den gesamten Trainingsdatensatz) werden 95.53% der 10000 Ziffern im (unabhängigen) Testdatensatz korrekt klassifiziert. Aufgrund der zufälligen Initialisierung der Netzwerk-Parameter können die konkreten Werte etwas abweichen.

(c) Verwenden Sie nun das Netzwerk, um mindestens zwei Ihrer eigenen handschriftlichen Ziffern zu klassifizieren. [2 Punkte]

Hinweis: Scannen oder fotografieren Sie Ihre Handschrift ab, wählen Sie dann einen Ausschnitt mit einer einzelnen Ziffer in dem entsprechenden Bild aus, z.B. mittels Gimp (https://www.gimp.org) oder einem ähnlichen Bildbearbeitungsprogramm, und skalieren Sie anschließend den Ausschnitt auf 28×28 Pixel. In der Datei `image_tools.py` findet sich die Hilfsfunktion `read_image_as_matrix(filename)`, mit der Sie das Bitmap-Bild Ihrer Ziffer als NumPy-Matrix einlesen können. Die Klassifizierung durch das Netzwerk erhalten Sie schließlich mittels `np.argmax(net.output(img.reshape((1,784))))`, wobei die Variable `img` das Bild speichert und hier der `reshape`-Befehl eine Zusatzdimension für einen minibatch mit nur einem Element hinzufügt.

Aufgabe 10 (Cross-entropy Kostenfunktion)
Die Cross-entropy Kostenfunktion für ein einzelnes Trainingspaar (x, y) lautet

$$C_{x,y} = -\sum_j \Big[y_j \log\big(a_j^L\big) + \big(1 - y_j\big) \log\big(1 - a_j^L\big) \Big], \tag{1}$$

wobei a^L den Ausgabevektor des Netzwerks bezeichnet (Ausgabeschicht L) und $\log$ den natürlichen Logarithmus. Das Netzwerk verwende die Sigmoid-Aktivierungsfunktion $\sigma(z) = 1/(1 + \mathrm{e}^{-z})$ in allen Schichten.

(a) Im Folgenden soll die spezielle Form (1) der Cross-entropy Kostenfunktion begründet werden. Zur verbesserten "Lernfähigkeit" sollte sich der Faktor $\sigma'(z_j^L)$ im Term

$$\delta_j^L := \frac{\partial C_{x,y}}{\partial a_j^L}\, \sigma'(z_j^L)$$

innerhalb der Backpropagation-Gleichungen für die Ausgabeschicht herauskürzen, und δ_j^L sollte umso größer sein, je stärker a^L von y abweicht, was den Ansatz $\delta_j^L = a_j^L - y_j$ nahelegt. Leiten Sie hieraus eine Differentialgleichung für $C_{x,y}$ (als Funktion von a^L) her, und zeigen Sie, dass diese von Gl. (1) gelöst wird. [3 Punkte]

(b) Eine Verallgemeinerung von Gl. (1) ist die Cross-entropy zweier diskreter Wahrscheinlichkeitsverteilungen p und q (wobei wir $p_i > 0$ und $q_i > 0$ für alle i annehmen):

$$H(p, q) = -\sum_i p_i \log(q_i).$$

[1] https://tu-dresden.de/mn/math/wir/mendl/studium/courses/mosim_2018_wise

Diese lässt sich auch als $H(p,q) = H(p) + D_{\mathsf{KL}}(p \parallel q)$ schreiben, wobei

$$H(p) = -\sum_i p_i \log(p_i)$$

die Entropie der Verteilung p ist, und

$$D_{\mathsf{KL}}(p \parallel q) = \sum_i p_i \log \frac{p_i}{q_i}$$

die Kullback-Leibler-Divergenz. Beweisen Sie die sogenannte Gibbs-Ungleichung: $D_{\mathsf{KL}}(p \parallel q) \geq 0$, mit Gleichheit genau für $p = q$. [3 Punkte]

Hinweis: Für alle $x > 0$ gilt $-\log(x) \geq (1-x)$, mit Gleichheit genau für $x = 1$. (Diese Aussage können Sie als bekannt voraussetzen.) Wenden Sie diese Aussage auf $\log(p_i/q_i) = -\log(q_i/p_i)$ an, und nutzen Sie die Normalisierung der Wahrscheinlichkeitsverteilungen ($\sum_i p_i = 1$, $\sum_i q_i = 1$) aus.

Tutoraufgabe 5 (Backpropagation für komplexwertige Netze)
In dieser Aufgabe sollen die Gleichungen des Backpropagation-Algorithmus auf komplexwertige neuronale Netze verallgemeinert werden, d.h. w^ℓ, b^ℓ, z^ℓ, a^ℓ sind jetzt komplexe Matrizen bzw. Vektoren. Wir nehmen an, dass die Aktivierungsfunktion in einer offenen Menge $U \subset \mathbb{C}$ holomorph ist, was ja insbesondere auf die Sigmoid-Funktion $\sigma(z) = 1/(1+\mathrm{e}^{-z})$ zutrifft. Die Kostenfunktion $C_{x,y}$ ist weiterhin reellwertig (so dass deren Minimierung überhaupt Sinn ergibt), und somit nicht komplex differenzierbar. Deshalb fassen wir jetzt die Ableitungsoperatoren als *Wirtinger-* oder *Dolbeault*-Operatoren auf:

$$\frac{\partial}{\partial z} := \frac{1}{2}\left(\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right), \quad \frac{\partial}{\partial \overline{z}} := \frac{1}{2}\left(\frac{\partial}{\partial x} + i\frac{\partial}{\partial y}\right)$$

mit $z = x+iy$, $x, y \in \mathbb{R}$. Wir benötigen noch die folgende Kettenregel, die sich durch direktes Nachrechnen verifizieren lässt:

$$\frac{\partial}{\partial z}(g \circ f) = \left(\frac{\partial g}{\partial w} \circ f\right) \cdot \frac{\partial f}{\partial z} + \left(\frac{\partial g}{\partial \overline{w}} \circ f\right) \cdot \frac{\partial \overline{f}}{\partial z}.$$

Hinweis: Die $\mathbb{C}$-lineare Multiplikation $w \mapsto z \cdot w$ mit einer vorgegebenen komplexen Zahl $z = x+iy$ lässt sich durch Identifikation $\mathbb{C} \cong \mathbb{R}^2$ auch als $\mathbb{R}$-lineare Abbildung auffassen. Anwendung auf $1 = (1,0)$ und $i = (0,1)$ ergibt die Matrixdarstellung

$$A_z = (z|iz) = \begin{pmatrix} x & -y \\ y & x \end{pmatrix}. \tag{2}$$

Umgekehrt ist eine reelle 2×2-Matrix $A = (z_1|z_2)$ mit Spaltenvektoren $z_1, z_2 \in \mathbb{C}$ also von der Form (2), wenn $z_2 = iz_1$ bzw. $z_1 + iz_2 = 0$.

Eine Funktion $f : U \subset \mathbb{C} \to \mathbb{C}$ ist bekanntlich genau dann in $z \in U$ komplex differenzierbar, wenn

$$f(w) = f(z) + f'(z) \cdot (w - z) + o(w - z) \qquad (w \to z). \tag{3}$$

Identifizieren wir $\mathbb{C} \cong \mathbb{R}^2$, dann ist f (als Funktion $U \subset \mathbb{R}^2 \to \mathbb{R}^2$) genau dann reell differenzierbar, wenn

$$f(w) = f(z) + Df(z) \cdot (w - z) + o(w - z) \qquad (w \to z).$$

Hierbei ist $Df(z) \in \mathbb{R}^{2\times 2}$ die Jakobi-Matrix. Die stärkere Bedingung "komplex differenzierbar" bedeutet also, dass das Matrix-Vektor-Produkt $Df(z) \cdot (w-z)$ als Multiplikation zweier komplexer Zahlen aufgefasst werden kann. Zerlegen wir $Df = (\partial_x f | \partial_y f)$, dann ist dies laut obiger Diskussion äquivalent zu den Cauchy-Riemann'schen Differentialgleichungen

$$\frac{\partial f}{\partial \overline{z}} \equiv \frac{1}{2}\left(\partial_x f + i\partial_y f\right) = 0,$$

wobei $\partial/\partial\overline{z}$ der (konjugierte) Wirtinger-Operator ist. Die komplexe Ableitung ist dann $f'(z) = \partial f(z)/\partial z$.

Als direkte Folgerung erhält man für f komplex differenzierbar in z:

$$\frac{\partial \overline{f}}{\partial z} = \frac{1}{2}\left(\partial_x \overline{f} - i\partial_y \overline{f}\right) = \overline{\frac{1}{2}\left(\partial_x f + i\partial_y f\right)} = \overline{\left(\frac{\partial f}{\partial \overline{z}}\right)} = 0. \tag{4}$$

(a) Berechnen Sie $\partial C_{x,y}/\partial a_j^L$ für die quadratische Kostenfunktion $C_{x,y} = \frac{1}{2}\|a^L - y\|^2$, wobei $\partial/\partial a_j^L$ die Wirtinger-Ableitung bezüglich a_j^L ist.

(b) Leiten Sie (formal identisch zum reellwertigen Fall) den Zusammenhang

$$\frac{\partial C_{x,y}}{\partial a_k^{\ell-1}} = \sum_j \frac{\partial C_{x,y}}{\partial a_j^\ell}\, \sigma'(z_j^\ell)\, w_{jk}^\ell$$

für alle k und $\ell \leq L$ her, wobei σ' die komplexe Ableitung der Aktivierungsfunktion bezeichnet.

(c) Verifizieren Sie ebenso

$$\frac{\partial C_{x,y}}{\partial w_{jk}^\ell} = \frac{\partial C_{x,y}}{\partial a_j^\ell}\, \sigma'(z_j^\ell)\, a_k^{\ell-1} \qquad \text{sowie} \qquad \frac{\partial C_{x,y}}{\partial b_j^\ell} = \frac{\partial C_{x,y}}{\partial a_j^\ell}\, \sigma'(z_j^\ell).$$

(d) Wie kann man nun den Gradienten der Kostenfunktion bezüglich des Real- und Imaginärteils von w_{jk}^ℓ bzw. b_j^ℓ erhalten?