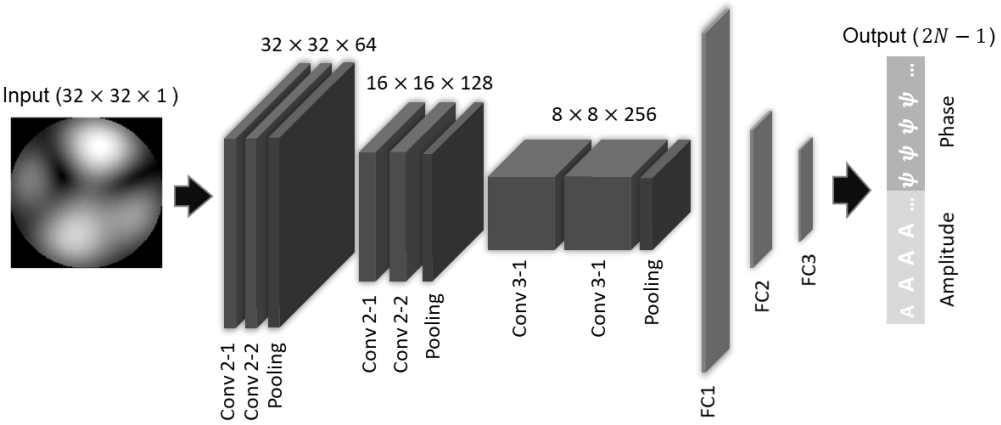


Praktikum Neuronale Netze in der Bildverarbeitung

Versuchsanleitung

Modendekomposition bei einer Multimodefaser

Stefan Rothe, Tom Glosemeyer, Qian Zhang, Jürgen Czarske

17. Dezember 2021

Inhaltsverzeichnis

1 Einleitung **1**

2 Grundlagen **1**
2.1 Einführung zur Modendekomposition 1
2.2 Architekturen 3
2.3 Erstellung von Trainingsdaten für die Modendekomposition 5
2.4 Transfer Learning 7
2.5 Quantitative Bewertung zur Leistungsfähigkeit 8

3 Fragen zur Versuchsvorbereitung **8**

4 Aufgaben zur Versuchsdurchführung **8**

1 Einleitung

Wie Denken wir? - Das menschliche Individuum ist nicht nur in der Lage, durch seine Sinnesorgane die Umgebung wahrzunehmen. Es kann auch mit dieser interagieren, beispielsweise durch Bewegung. Während bei der Wahrnehmung die Umgebungsgrößen in Reize umgewandelt werden, wandelt ein Muskel diese Reize in Bewegung um. Das „Denken“ bezeichnet in dieser Darstellung die Umwandlung und Verteilung der Reize zwischen sensorischer und aktorischer Ebene durch ein komplexes Netzwerk aus Nerven [2]. In der allgemeinen Auffassung wird die kleinste Einheit dieses Nervensystems als Neuron bezeichnet. Es wird angenommen, dass dieses verschiedene Reize aufnehmen, gegeneinander wichten und an weitere Neuronen weitergeben kann. Stellt man sich die Reize als eine einfache Zahl vor, lässt sich der Sachverhalt leicht in einem mathematischen Modell darstellen und bildet die Grundlage für die Systeme künstlicher Neuronaler Netze. Diese und alle weiteren grundlegenden Betrachtungen können in [1] nachgelesen werden.

Ziel des dritten Praktikumversuchs Im dritten Teil des Praktikums werden mithilfe von Bildbverarbeitungstechniken basierend auf neuronalen Netzen (NN) phyikalsiche Parameter elektromagnetischer Felder bestimmt. Die komplexen Felder einer mehrmodigen Glasfaser (Multimodefaser, MMF), können als Überlagerung einzelner Grundschwingungen interpretiert werden. Diese werden auch als Eigenfunktionen des physikalsichen Modells des Wellenleiters betrachtet und als Moden bezeichnet. Die Feldverteilungen der Moden werden mit den Maxwell-Gleichungen errechnet und sind für eine vorliegende MMF stets bekannt. Jede beliebige Kombinationen der ausbreitungsfähigen Moden erzeugen eine der möglichen Feldverteilungen für die MMF. Kennt man die Parameter der vorliegenden MMF und daher auch die Basis ausbreitungsfähiger Moden, kann jede MMF-Feldverteilung wieder zurück in die orthogonale Modenbasis zerlegt, bzw. „dekomponiert“ werden. Das Ergebnis sind komplexe Gewichte, die den Anteil jeder Mode an der jeweiligen Feldverteilung repräsentieren. Als Analogon für die Modendekomposition bietet sich die Fourierreihenanalyse für harmonische Signale an, bei der jedes analoge Signal als Superposition von Sinus- und Kosinusschwingungen analysiert werden kann. Die Modendekomposition hingegen, kann mithilfe von Bildverarbeitungsalgorithmen gelöst werden, da die 2D-Feldverteilungen pixelierte Bildinformationen liefern. Interessanterweise ist es möglich ein einfaches Intensitätsbild (Kamerabild) einer MMF-Lichtverteilung mithilfe eines NN zu analysieren, dass die komplexen Gewichte der Moden geschätzt werden. Besonders hierbei ist, dass beide komplexen Feldkomponenten, Amplitude und Phase, auf Basis des reellen Kamerabildes extrahiert werden können. Eine wesentliche Herausforderung ist das Design der Trainingsdaten für ein funktionierendes NN. Die Hürde stellen Mehrdeutigkeiten in der Phase dar, da auch die elektromagnetischen Felder Periodizitäten aufweisen. Es ist zu beachten, dass jedem Bild ausschließlich ein komplexer Vektor zugeweisen wird. Zusätzlich steigt die Komplexität des zu lösenden Problems der Modendekomposition, je mehr Moden in der MMF ausbreitungsfähig sind. Hierfür wird die Methode „Transfer Learning“ vorgestellt. Mithilfe von Transfer Learning können bereits fertig trainierte Netze, deren Parameter eingestellt sind, an neue, teilweise erweiterte Problemstellungen angepasst werden. Die bereits trainierten Netze bilden damit den Ausgangspunkt für das neue Training. Hiermit sinkt die Anforderung an den Trainingsvorgang im Vergleich zu zufällig verteilten Parameterkonfikurationen zu Beginn eines Trainings. Die Hoffnung hierbei ist, dass mit Transfer Learning schrittweise aber möglichst niedrigschwellig, komplexe Problemstellungen erlernt werden können.

2 Grundlagen

2.1 Einführung zur Modendekomposition

Lichtfelder, die sich durch eine Multimodefaser (MMF) ausbreiten, können mathematisch und physikalisch beschrieben werden. Je nach Spezifikation des einkoppelnden Lichtstrahls und Materialbeschaffenheit des Wellenleiters, können verschiedene Lichtmuster in der MMF entstehen. Die grundlegende physikalische Beschreibung der Einzelfelder erfolgt durch die Lösung der Maxwell'schen Gleichungen. Die Einzelmuster bilden dabei eine sog. orthogonale Basis und werden als *Moden* bezeichnet. Aufgrund der Orthogonalität zueinander lässt sich ein Einzelmuster nicht

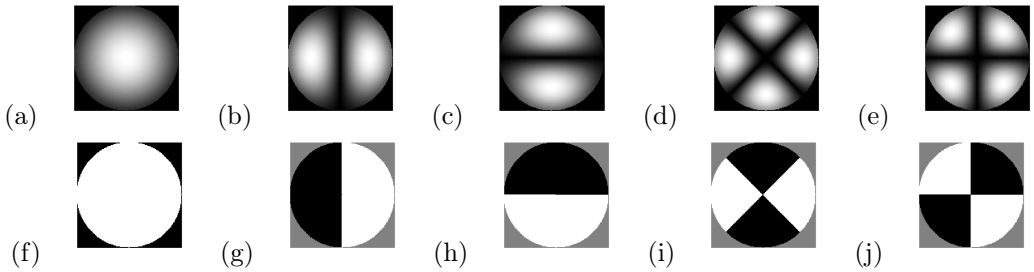


Abbildung 1: Lichtfeldverteilungen der einzelnen ausbreitungsfähigen Moden mit den Parametern: $\lambda = 750\,\text{nm}$, NA= 0,1 und $a = 5\,\mu\text{m}$. Es können sich insgesamt $N = 5$ Moden ausbreiten. Abbildungen (a)-(e) zeigen Amplituden-, während die Abbildungen (f)-(j) die zugehörigen Phasenverteilungen darstellen. Die Amplitudenbilder sind zu Darstellungszwecken jeweils auf 1 normiert. In den Phasenbildern implizieren die schwarzen Bereiche einen relativen Phasenunterschied von π zu den weißen Bereichen.

aus einer Linearkombination der anderen Muster bilden. Ihr Skalarprodukt ist dadurch null. Das gleiche Phänomen ist bei Sinus- und Kosinusfunktionen in der Signaltheorie zu beobachten. Sinus und Kosinus bilden eine orthogonale Basis für jedes analoge Zeitsignal, können jedoch nicht durch Linearkombination ineinander überführt werden. Jedoch kann durch Einsatz der Fourieranalyse jedes beliebige Signal in eine Kombination aus Sinus- und Kosinussignalen zerlegt werden. Der gleiche Ansatz wird bei der Modendekomposition bei der MMF verfolgt. Jedes beliebige Lichtfeld E_{MMF} kann in eine Kombination aus den Moden E_i zerlegt, bzw. dekomponiert werden:

$$E_{\text{MMF}} = \sum_{i}^{N} k_i E_i. \tag{1}$$

Dabei bezeichnet der Faktor k_i das Gewicht der jeweiligen Mode und repräsentiert den Anteil jeder Mode am MMF-Feld. Da eine kohärente Lichtausbreitung angenommen wird, sind k_i komplexe Gewichte und bestehen, wie die Moden E_i, aus Amplitude ρ und Phase φ: $k_i = \rho_i e^{i\varphi_i}$. Glücklicherweise sind durch die Herstellerangaben von MMF und eingesetztem Laser sämtliche Systemparameter bekannt, wodurch die Anzahl Moden N vorab bekannt ist. Diese setzt sich aus dem sog. V-Parameter zusammen, der ein dimensionloser Parameter zur charakterisierung der Lichtübertragung durch eine Glasfaser ist:

$$V = \frac{2\pi}{\lambda}\text{NA}a. \tag{2}$$

Gegenwärtig erhältliche Laser weisen ein überaus schmalbandiges Spektrum auf, sodass die angegebene mittlere Wellenlänge λ als einfacher Skalar angenommen werden kann. Die Produktion von MMF weist ebenfalls äußerst geringe Toleranzen bei der numerischen Apertur NA (siehe Anleitung zu Versuch 2) und dem Kernradius a der MMF auf. Liegt der V-Parameter einer beliebigen Glasfaser oberhalb des Wertes $2,4$, so kann sich mehr als nur eine Mode ausbreiten und die Faser gilt als multimodal. Die Anzahl ausbreitungsfähiger Moden lässt sich im Allgemeinen mit $N \approx \frac{V^2}{2}$ ausdrücken, wobei zwei mögliche Polarisationsrichtungen mit identischen Moden berücksichtigt werden. Sind alle Parameter aus Gleichung (2) bekannt, können die Moden, die sich durch die vorliegende MMF ausbreiten können, mithilfe der Maxwell-Gleichungen ermittelt werden. In Abbildung 1 sind die 5 Feldkoponenten der 5 ausbreitungsfähigen Moden einer MMF dargestellt. In Abhängigkeit von Temperaturschwankungen, Krümmungen oder Verdrillungen der MMF, können sich beliebige Kombinationen der in Abbildung 1 gezeigten Moden ergeben. Obwohl die Anteile der Feldzusammensetzung durch die Berechnung der Moden bekannt ist, ist die zeitliche Entwicklungen umgebender Einflussbedingungen praktisch unvorhersehbar, weshalb die augenblickliche Zusammensetzung des Feldes nicht berechnet werden kann. Aus diesem Grund ist es für eine Charakterisierung der Feldzusammensetzung erforderlich zu messen. Ziel ist es, die komplexen Ge-

wichte k_i aus Gleichung (1) zu ermitteln. In Abbildung 2 ist das Prinzip der Modendekomposition dargestellt.

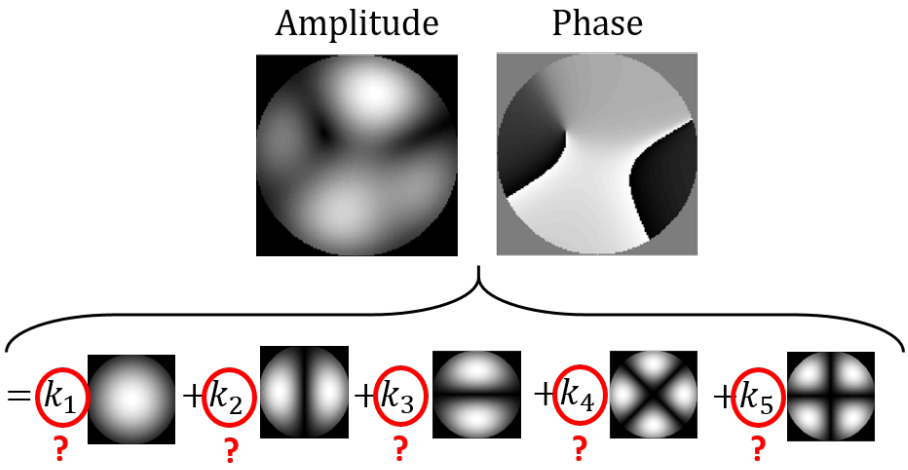


Abbildung 2: Prinzip der Modendekomposition. Ein beliebiges Lichtmuster einer MMF setzt sich aus der komplexen Basis der Moden zusammen. Ziel der Modendekomposition ist es, die komplexen Gewichte der Moden zu bestimmen und die vorliegende Zusammensetzung zu charakterisieren.

Sind beide komplexen Feldkomponenten, Amplitude und Phase, durch eine Messung (bspw. digitale Holographie) bekannt, können die komplexen Gewichte analytisch berechnet werden. Üblicherweise kann die Amplitude eines Lichtfeldes durch die Beleuchtung eines Kamerasensors ermittelt werden. Der Kamerasensor liefert die Intensität des Feldes, welche das Quadrat der Amplitude ist. Im Allgemeinen sind jedoch Messaufbauten zur Bestimmung der Phase vergleichweise aufwändig, sowie rauschanfällig. Daher ist es wünschenswert eine Methode zur Schätzung der Phase zu ermöglichen, die auf Basis eines simplen Kamerabildes (Intensität) funktioniert. Hierfür soll ein neuronales Netz trainiert werden, welches als Eingabe ein Intensitätsbild verwendet, um dann die komplexen Gewichte, sowohl Amplitude als auch Phase, vorherzusagen.

2.2 Architekturen

In diesem Abschnitt sollen zwei Architekturen vorgestellt werden, mit denen Regressionsaufgaben gelöst werden können. Im Falle dieses Praktikumsversuchs ist dies die Modendekomposition aus dem Intensitätsbild. Zum einen können erneut die aus den früheren Versuchen bekannten MLPs verwendet werden, aber auch die weit verbreiteten Very Deep Convolutional Neural Networks (VGG) werden vorgestellt.

- **MLP**
  Die Netzarchitektur Multilayer Perceptron wurde bereits im ersten und zweiten Praktikumsversuch zur Klassifizierung von Ziffern und der Rekonstruktion von Ziffern aus den Specklemustern einer MMF eingesetzt. Es ist wichtig zu beachten, dass die Dimensionalität des output layer mit den Daten übereinstimmt. Der Beispielcode für ein MLP lautet wie folgt:

```
fullyConnectedLayer(input_size*input_size,'Name','fc1')
leakyReluLayer('Name', 'relu1')
fullyConnectedLayer(input_size*input_size,'Name','fc2')
leakyReluLayer('Name','relu2')
fullyConnectedLayer(output_size,"Name","fc_output")
```

- **VGG**
  VGG ist ein neuronales Faltungsnetzmodell, das von Simonyan und Zisserman von der Visual

Geometry Group (VGG) an der Universität Oxford eingeführt wurde [4]. VGG hat verschiedene Konfigurationen, z. B. 11-layers, 13-layers, 16-layers, 19-layers. Die Einführung erfolgt am Beispiel des VGG-9, siehe Abbildung 3. das VGG ist in 4 Blöcke unterteilt. Der erste Block besteht aus zwei Convolutional Layer mit Kernel 3×3. Dies ist das erste Mal, dass ein kleiner Convolutional-Kernel in einem neuronalen Netz verwendet wurde. Das rezeptive Feld von zwei Convolutional Layer mit Kernel 3×3 entspricht dem eines Convolutional Layer mit Kernel 5×5, aber mit einer geringeren Anzahl von Parametern. Aus Gründen der Vereinfachung wird die ReLU-Aktivierungsfunktion in der Abbildung nicht dargestellt. Das letzte Layer des ersten Blocks ist ein 2×2 Max-Pooling-Layer mit einer Schrittweite von 2, das die Größe der Feature Map um den Faktor 2 reduziert. Der zweite und dritte Block ist ähnlich aufgebaut wie der erste Block, jedoch verdoppelt sich die Anzahl der Kanäle. Der vierte Block des VGG besteht aus 2 Fully Connected Layern, die alle zuvor gesammelten Merkmalsinformationen integrieren können. In diesem Versuch kann die Größe des Ausgangs dieser Layer auf 256 und 128 eingestellt werden. Das output layer ist ebenfalls ein Fully Connected Layer mit einer Ausgabegröße von $2N - 1$ (siehe Abschnitt 2.3 zur Dimension des output layers). In Anbetracht der Tatsache, dass alle Ausgangswerte zwischen 0 und 1 liegen, kann eine Aktivierungsfunktion, z.B. Sigmoid, hinzugefügt werden. Wird diese verwendet verwendet, so sind ihre Nachteile zu beachten. Am offensichtlichsten ist das sog. *gradient vanishing*. Wenn sich die Aktivierungsfunktion dem Sättigungsbereich nähert, ist die Änderung zu langsam und die Ableitung nähert sich 0.

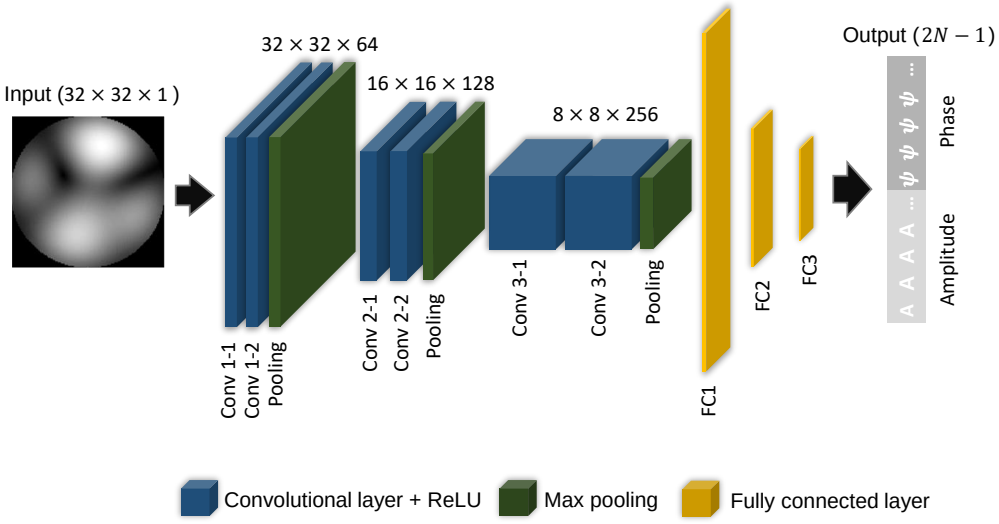


Abbildung 3: Architektur des VGG für die Durchführung einer Modendekomposition.

Tatsächlich enthält das ursprüngliche VGG 5 Convolution Blocks. Jeder Block besteht aus 1 bis 4 Convolutional Layers. Die Anzahl der Kanäle beträgt 64, 128, 256, 512, 512. Die Dimension des Fully Connected Layers beträgt 4096. Darüber hinaus wird ein Dropout Layer verwendet, um das Problem des Overfittings möglichst zu kompensieren. Die Struktur des VGG kann daher an die Komplexität des Problems und den Stand der Hardware angepasst werden.

Der Beispielcode zum Erstellen eines VGG in Matlab ist hier angegeben:

```
% ein Block
convolution2dLayer(3,64,'Name',"conv1_1","Padding","same")
reluLayer('Name',"relu1_1")
convolution2dLayer(3,64,'Name',"conv1_2","Padding","same")
reluLayer('Name',"relu1_2")
maxPooling2dLayer(2,"Stride",2,'Name',"pooling1")
```

```
...
...
fullyConnectedLayer(256,'Name',"fc1")
fullyConnectedLayer(128,'Name',"fc2")
fullyConnectedLayer(output_size,'Name',"fc_output")
```

2.3 Erstellung von Trainingsdaten für die Modendekomposition

Das Ziel ist es, mithilfe der Netzarchitekturen in Abschnitt 2.2 eine Modendekomposition durchzuführen. Eingangsseitig werden reine Intensitätsbilder von Feldern aus Gleichung (1) in das Input Layer geladen. Wichtig hierbei ist, dass die Phaseninformation nicht in das Netz geladen wird. Das Output Layer besteht aus den Amplituden- und Phasenkomponenten (ρ und φ) der komplexen Gewichte ($k_i = \rho_i e^{i\varphi_i}$). Besonders hierbei ist, dass die Phaseninformation auf Basis einer Intensitätsverteilung extrahiert wird. Die für die Modendekomposition verwendeten Felder E_{MMF} bestehen dabei aus Kombinationen der ausbreitungsfähigen Moden. Es können sich bei einer MMF, die 5 Moden führen kann, bspw. die Moden aus Abbildung 1 ausbreiten. Intuitiv liegt es zur Erstellung von Trainingsdaten nahe, eine große Anzahl zufälliger Kombinationen der Moden aus Abbildung 1 zu erzeugen. Der Trainingsdatensatz besteht dann aus der Intensitätsverteilung, welche als Bild gespeichert wird und dem zugehörigen Label-Vektor. Hier ist zu beachten, dass eine Regression durchgeführt wird. In MATLAB können zufällige, gleichverteilte Zahlen mit der Funktion `rand()` erzeugt werden. Allerdings sind die Felder der MMF physikalische Systeme, in denen stets das Prinzip der Energieerhaltung gilt. Energieerhaltung kann für simulativ erzeugte Gewichte berücksichtigt werden, indem jeder Vektor, welcher mit `rand()` kreiert wird, durch seine Norm (MATLAB-Befehl `norm()`) geteilt wird. Es gilt dann:

$$\sum_{i}^{N} |\rho_i|^2 = 1. \tag{3}$$

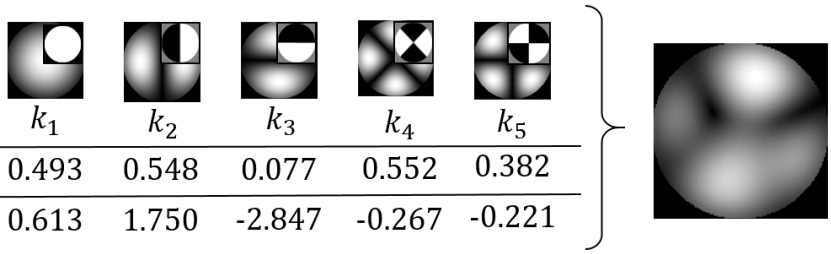
Die Komponenten der Phase können zunächst ebenfalls mittels `rand()` erstellt werden. Es ist darauf zu achten, dass die Phase im Intervall $[-\pi; \pi)$ liegen kann. Ein Vektor für eine Kombination aus 5 Moden kann schließlich folgendermaßen erstellt werden:

```
rho=rand(1,5);
rho_n=rho/norm(rho);
k_i=rho_n.*exp(1i*rand(1,5)*2*pi);
```

ein Beispiel für einen solchen Vektor ist in Abbildung 4 dargestellt. Dieser Vektor enthält nun die komplexen Gewichte, mit denen die komplexen Feldverteilungen aus Abbildung 1 multipliziert werden, um ein MMF-Feld (Gleichung (1)) zu bilden. Der dort gezeigte Vektor muss für das Training weiter verarbeitet werden und kann noch nicht als Label-Vektor verwendet werden.

Für ein erfolgreiches Training ist es erforderlich, Phasenmehrdeutigkeiten zu eliminieren. Da komplexe Elemente betrachtet werden, treten Periodizitäten in der Phase auf. Periodizitäten können sowohl bei harmonischen Signalen als auch in komplexen Feldern entstehen und führen dazu, dass Information nicht eineindeutig zugeordnet werden kann. In diesem Fall spricht man von Mehrdeutigkeiten, die beim Trainieren neuronaler Netze zu signifikanten Problemen führen. In dem hier betrachteten Fall ist es wichtig, dass ein Bild einer Modenkombination einem einzigen Vektor zugeordnet werden kann. Es existieren zwei Grundlegende Quellen von Mehrdeutigkeiten bei der Erzeugung von Trainingsdaten, die berücksichtigt werden müssen:

1. Bei der Überlagerung komplexer Felder ist nur der *relative* Phasenunterschied der Moden untereinander bedeutend. Das heißt, wird ein beliebiger Vektor mit Gewichten k mit einem Offset in der Phase versehen, entsteht eine identitische Intensitätsverteilung. Dieses Phänomen muss bei der Erstellung von Trainingsdaten berücksichtigt werden, da sonst mehrere identische Bilder unterschiedlichen Label-Vektoren zugeordnet werden. Aus diesem Grund wird ein Bezugspunkt verwendet, zu dem der relative Phasenunterschied betrachtet wird. Es



	k_1	k_2	k_3	k_4	k_5
ρ_i	0.493	0.548	0.077	0.552	0.382
φ_i	0.613	1.750	-2.847	-0.267	-0.221

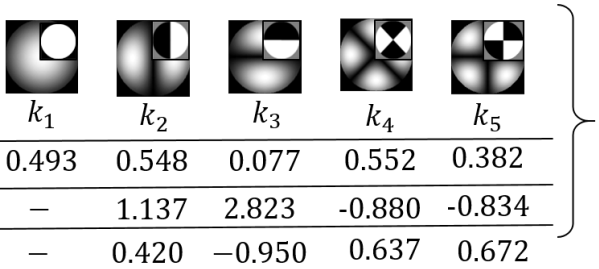
Abbildung 4: Beispiel für einen Vektor zur Erzeugung komplexer Gewichte für die Modendekomposition. Der Vektor muss noch weiter verarbeitet werden und kann noch nicht als Label-Vektor verwendet werden. Es wurden 5 komplexe Gewichte, welche der Energieerhaltung (Gleichung (3)) unterliegen, gebildet. Die Phase ist in rad angegeben. Mit den Gewichten können die komplexen Feldverteilungen (Abbildung 1) multipliziert und anschließend aufsummiert werden, wodurch das Feld E_{MMF} auf der rechten Seite entsteht. Für das Training wird von E_{MMF} ausschließlich die Amplitude, bzw. Intensität verwendet.

wird hierfür bspw. die Phase der ersten Mode auf null gesetzt und nur der relative Unterschied hierzu betrachtet. Das heißt, dass nach Erstellen des Vektors k die Phasenelemente um die Phase der ersten Mode $\arg(k_1)$ subtrahiert werden müssen. Der Label-Vektor im Output Layer erhält daher die Dimension $2N-1$, da die Phase der ersten Mode ignoriert werden kann.

Dieses Phänomen tritt ebenfalls bei harmonischen Signalen auf. Stellen Sie sich vor, dass zwei harmonische Signale überlagert werden $y(t) = a_0 \cdot \sin(\omega_1 t) + b_0 \cdot \sin(\omega_2 t)$. Das resultierende Signal $y(t)$ weist auf der Zeitskala stets ein gleichbleibendes Verhalten auf, wenn beide Signale um denselben Offset in der Phase verschoben werden. Erst, wenn die Signale *relativ* zueinander verschoben werden, ändert sich auch die Resultierende. In Analogie hierzu wird für den Label-Vektor das erste Signal in den Ursprung gesetzt und lediglich der *relative* Unterschied zum Bezugspunkt betrachtet.

2. Bei komplexen Feldern ergibt sich die Intensität aus dem Betragsquadrat der Amplitude, welches eine Multiplikation des Feldes E mit seiner konjugiert komplexen E^* Version ist: $I \sim |E|^2 = E \cdot E^*$. Aus diesem Grund ist die Intensität eines Feldes gleich der Intensität des konjugierten Feldes. Erneut kann dies für das Netz zu Mehrdeutigkeiten führen, da zwei identische Intensitätsbilder unterschiedliche Label-Vektoren zugewiesen bekommen können. Für die einzelnen Moden gilt daher eine Eineindeutigkeit ausschließlich im Phasenbereich $[0;\pi)$. Da jedoch die Phasenwerte der Gewichte k im Bereich $[-\pi;\pi)$ vorliegen, und dementsprechend negative Imaginärteile auftreten dürfen, wird für den Label-Vektor der Kosinus der Phase der Gewichte k verwendet. Dieser bildet alle Phasenwerte im Bereich $[-\pi;\pi)$ auf den Bereich $[-1;1]$ ab, wobei der Kosinus die Eigenschaft besitzt: $\cos(\arg(k_i)) = \cos(\arg(k_i^*))$. Somit sind die Phasenwerte der Gewichte k_i und ihre komplex konjugierte Version k_i^* auf den gleichen Bereich abgebildet und Mehrdeutigkeiten eliminiert.

Werden beide Maßnahmen zur Vorbeugung von Phasenmehrdeutigkeiten berücksichtigt, ändern sich die durch Zufallszahlen kreierten Phasenwerte in den komplexen Gewichten aus Abbildung 4. Die korrigierte Version des Vektors inkl. der Kosinuswerte sind in Abbildung 5 gezeigt. Nach Berücksichtigung von Maßnahme 1 (Bildung Bezugspunkt Mode 1 und relativer Phasenunterschied), ändert sich die resultierende Intensität im Vergleich zu Abbildung 4 nicht. Die Kosinus-Werte der Phasenelemente der einzelnen Gewichte werden schließlich zusammen mit den Amplitudenwerten als Label-Vektoren verwendet. Ein Label-Vektor für N Moden, bzw. ein Vektor mit komplexen Gewichten hat dementsprechen $2N-1$ Elemente. Im Allgemeinen wird die Leistungsfähigkeit eines Trainings verbessert, wenn alle Elemente des Vektors im gleichen Intervall liegen. Häufig wird versucht, das Intervall $[0;1]$ zu verwenden, weshalb die Kosinus-Werte aus dem Intervall $[-1;1]$ auf den Bereich $[0;1]$ skaliert werden sollten.



	k_1	k_2	k_3	k_4	k_5
ρ_i	0.493	0.548	0.077	0.552	0.382
φ_i	–	1.137	2.823	-0.880	-0.834
$\cos(\varphi_i)$	–	0.420	−0.950	0.637	0.672

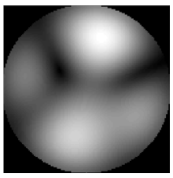


Abbildung 5: Korrigierte Version der Vektoren k_i aus Abbildung 4. Mehrdeutigkeiten sind mit den beiden Vorkehrungen aus Abschnitt 2.3 eliminiert. Die Phase des Gewichts der ersten Mode k_1 wird auf null gesetzt und der relative Phasenunterschied der übrigen Moden zu diesem Bezugspunkt betrachtet. Dementsprechend müssen die Phasenwerte der anderen Gewichte k_{2-5} mit diesem Wert subtrahiert werden. Man beachte, dass nach Korrektur der Phase im Gewicht k_3 (vgl. Abbildung 4) die untere Intervallgrenze bei $-\pi$ überschritten wurde und daher der resultierende Wert phasenrichtig bei der gegenüberliegenden Grenze einrückt. Neben den Amplitudenwerten werden die Kosinus-Werte der Phasenelemente als Label-Vektoren verwendet.

2.4 Transfer Learning

Transfer Learning (TL) ist eine Methode, die von Pan und Yang im Jahr 2009 vorgestellt wurde [3]. Bei TL wird ein bereits trainiertes NN zur Lösung neuer Probleme eingesetzt. Dazu wird der Lernfortschritt des bestehenden Modells übertragen. Daraus ergeben sich Vorteile wie: schnelleres Training, bessere Modellqualität, und geringerer Ressourcenverbrauch. Die Grundidee von TL besteht darin, die Tatsache auszunutzen, dass NNs in der Lage sind, dieselben Merkmale (z. B. Linien oder Punkte) aus Bildern zu extrahieren. Es bietet einen neuen Rahmen für das Training von groß angelegten tiefen neuronalen Netzen, die ein Problem mit hoher Komplexität lösen müssen. In diesem Praktikumsversuch wird TL verwendet, da das vorliegende Problem sehr einfach zu skalieren ist. Beispielsweise kann ein Netz, das zwischen 3 Moden unterscheiden kann, als Ausgangszustand für das Training eines Netzes für 5 Moden genommen werden. Oder ein NN, das mit einer kleinen Datenmenge trainiert wurde, kann als Ausgangspunkt verwendet werden, wenn die Datenmenge erhöht werden muss. Zu beachten ist, dass in Matlab das Transfer Learning nur für `dlnet` verwendet werden kann, da das SeriesNetwork (trainiert mit trainNetwork-Funktion) keinen Zugang zu den internen lernbaren Parameter hat. Das Verfahren ist wie folgt:

```
1  read LayerArray from trained dlnet & new dlnet
2  for all layers in LayerArray (exkl. Input & Output layer)
3      LayerProperties = LayerArray(index)
4      if LayerName match
5          read LearnableValue from trained dlnet
6          read LearnableValue from new dlnet
7          if size(LearnableValue) ~= 0
8              if size(LearnableValue) match
9                  LearnableValue_new_dlnet = LearnableValue_trained_dlnet
```

Bei TL ist es wichtig, dass die Struktur des Layers konsistent ist und dass die Dimensionalität der lernbaren Parameter konsistent ist. Die Eigenschaften des NN können wie folgt ausgelesen werden:

```
% Layer array
dlnet.Layer
% Network learnable parameters
dlnet.Learnables
% learnable parameters array
dlnet.Learnables.Value
```

2.5 Quantitative Bewertung zur Leistungsfähigkeit

- **Ermittelter Vektor**
  Um das Trainingsergebnis zu evaluieren, kann zunächst direkt der vorhergesagte Vektor mit dem Ground Truth verglichen werden. Wie schon im vorherigen Versuch kann dafür mit einem Validationsdatensatz z.B. die Wahrscheinlichkeitsverteilung des RMSE ermittelt werden, sodass das Ergebnis in einem Boxplot dargestellt werden kann.

- **Rekonstruktion basierend auf der Vorhersage**
  Alternativ ist es aber auch möglich, anhand der ermittelten Moden im Vektor zu versuchen, das ursprüngliche Bild mit dem Skript `mmf_rebuilt_image.m` zu rekonstruieren. Diese Rekonstruktion kann dann mit diesem Bild verglichen werden. Dafür kann z.B. die Korrelation verwendet werden, die ebenfalls aus dem vorherigen Versuch bekannt ist. Für einen Validierungsdatensatz kann hier eine Korrelationsverteilung bestimmt werden, die in einem Boxplot oder Scatterplot dargestellt wird.

3 Fragen zur Versuchsvorbereitung

Bevor Sie mit der Bearbeitung der Aufgaben starten, beantworten Sie als Vorbereitung bitte folgende Fragen:

1. Was sind Moden einer Faser und wie kann man diese berechnen?
2. Weshalb müssen Amplitudenelemente in einem Vektor mit komplexen Modengewichten normalisiert werden?
3. Wodurch können Mehrdeutigkeiten in der Phase eines komplexen Feldes auftreten?
4. Was ist Transfer Learning und wozu kann es hilfreich sein? Gehen Sie auf die Komplexität der zu lösenden Modendekomposition bei einer 3-Moden, bzw. einer 5-Moden MMF ein.
5. Beschreiben Sie die VGG Netzarchitektur. Was ist der Unterschied, bzw. der Vorteil der 3×3-Kernel Architektur, gegenüber der mit 5×5-Kernel?
6. Warum sollte ein Sigmoid-Layer am Ende der VGG-Netzarchitektur verwendet werden?

4 Aufgaben zur Versuchsdurchführung

Im Rahmen des Praktikums sollen folgende Aufgaben bearbeitet werden:

1. Benutzen Sie das MATLAB Skript `data_generation.m` zur Erzeugung des Datensatzes. Generieren Sie 10,000 komplexe Gewichte für eine 3-Moden MMF mit einem entsprechenden Code. Berücksichtigen Sie die Energieerhaltung für die Amplitudengewichte. Vermeiden Sie Mehrdeutigkeiten in der Phase (relativer Phasenunterschied wichtig!). Verwenden Sie die Amplitudengewichte und die Kosinus-Werte der Phasenelemente als Label-Vektoren. Denken Sie daran, dass die Dimension des Label-Vektors $2N - 1$ für eine N-Moden MMF ist. Teilen Sie die Label-Vektoren in Trainings-, Validierung- und Testdaten auf.
2. Verwenden Sie die Funktion `mmf_build_image.m`, um die zu den Label-Vektoren zugehörigen Bilddaten zu erzeugen. Füllen Sie die Lücken im Skript. Beutzen Sie die komplexen Feldverteilungen der 3 Moden aus `mmf_3modes_32.mat` zur Erzeugung der Feldkombinationen. Verwenden Sie den Befehl `abs()`, um die Amplituden der Feldverteilungen zu extrahieren. Skalieren Sie die Amplituden Feldverteilungen auf den Bereich [0,1]. Speichern Sie die Bilder in die jeweils richtigen Datensets für Training, Validierung und Test.

3. Erstellen Sie ein MLP zur Modendekomposition in dem Skript `training_seriesnet.m`. Achten Sie auf die benötigte Auflösung der Eingangsbilder und des Ausgangsvektors. Definieren Sie passende Hyperparameter für den Trainingsvorgang mit `trainingOptions`. Starten Sie anschließend den Trainingsvorgang mit `trainNetwork`. Wenn Sie eine CPU verwenden, können Sie die `'ExecutionEnvironment'` als `'parallel'` einstellen, um den Trainingsvorgang zu beschleunigen. Hiermit können mehrere CPU-Kerne an dem Trainingsvorgang beteiligt sein.

4. Evaluieren Sie die trainierten Netze mithilfe der Testdaten. Rekonstruieren Sie die Feldverteilungen auf der Grundlage der Vorhersageergebnisse des neuronalen Netzes mit dem Skript `mmf_rebuilt_image.m`. Innerhalb der Funktion werden automatisch die Amplitudengewichte direkt ermittelt, während der Phasenwert durch den Vergleich aller möglichen Kombinationen bestimmt werden muss. Füllen Sie die Lücken im Skript. Verwenden Sie den Korrelationskoeffizienten als Beurteilungskriterium. Die Kosinuswerte sollten zunächst auf das Intervall [-1,1] skaliert werden. Verwenden Sie den Befehl `arc()`, um die Phasengewichte zu ermitteln.

5. Visualisieren Sie die Ergebnisse in einem geeigneten Plot mithilfe des Korrelationskoeffizienten und berechnen Sie die Standardabweichung der Ergebnisse. Berechnen Sie die relative Abweichung der vorhergesagten Gewichte in Amplitude und Phase.

6. Wiederholen Sie die vorherigen Schritte für 10,000 Datenpaare und ein MLP für 5-Moden. Erstellen Sie dann ein VGG zur Modendekomposition für 3- und 5-Moden. Visualisieren Sie die Ergebnisse basierend auf den Testdaten.

7. Erstellen Sie ein `dlnet-type` VGG zur Modendekomposition für 3−Moden in dem Skript `training_dlnet.m`. Definieren Sie passende Hyperparameter für den Trainingsvorgang. Für das benutzerdefinierte Training steht Ihnen die Anleitung von Versuch 1 zur Verfügung. Das Training sollte ähnliche Ergebnisse liefern wie das VGG aus Aufgabe 6.

8. Trainieren Sie ein VGG für 5−Moden mit 10,000 Datenpaaren durch Transfer Learning (TL). Beachten Sie, dass dieses Training auf dem in Aufgabe 7 trainierten VGG für 3−Moden basieren soll. Visualisieren Sie das Ergebnis (Vorher/Nachher).

9. Erhöhen Sie die Trainingsdaten auf 50.000 Datenpaare für 5−Moden und trainieren Sie das VGG weiter. Verwenden Sie das in der Aufgabe 8 trainierten VGG als Startpunkt. Visualisieren Sie das Ergebnis (Vorher/Nachher).

References

[1] Charu C Aggarwal et al. "Neural networks and deep learning". In: *Springer* 10 (2018), pp. 978–3.

[2] Visible Body. *Gehirn und Nerven: 5 Schlüssel öffnen das Nervensystem.* `https://www.visiblebody.com/de/learn/nervous/system-overview`. [Online; accessed 11-November-2021]. 2021.

[3] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[4] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).