G

Grant ausgelegt

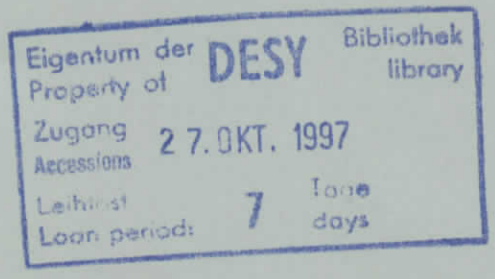
Interner Bericht
DESY F35D-97-10
September 1997

Identification of jets in deep inelastic e p scattering by neural networks

Jet-Identifizierung in der tiefunelastischen ep-Streuung mittels neuronaler Netze

von

M. Sievers



Jet-Identifizierung in der tiefunelastischen ep-Streuung mittels neuronaler Netze

Experimentelle Diplomarbeit
am Fachbereich Physik
der Universität Hamburg

Michael Sievers

September 1997

Jet-Identifizierung in der tiefunelastischen ep-Streuung mittels neuronaler Netze

Experimentelle Diplomarbeit
am Fachbereich Physik
der Universität Hamburg

Michael Sievers

September 1997

Inhaltsverzeichnis

1 Einleitung

Das Triggersystem des ZEUS-Detektors ist in drei Stufen aufgeteilt. Die erste Stufe muß mit einer Rate von maximal 1kHz Entscheidungen innerhalb von Mikrosekunden fällen, wobei keine Totzeit auftreten darf. Aus diesen Gründen ist sie als Hardware unter Verwendung einer pipeline realisiert. Die beiden folgenden Stufen reduzieren von 1kHz auf 100Hz bzw. von 100Hz auf 3-5Hz und können daher als Software realisiert werden. Bei der Suche nach Jets in der tiefunelastischen ep-Streuung wird die Datenmenge noch weiter auf ca. 10^{-4}Hz reduziert.

Das Problem, gewünschte Ereignisse von unerwünschten zu trennen, ist eines der Mustererkennung. Wegen des großen benötigten Reduktionsfaktors ist es ein schwieriges Problem. Es wird konventionell durch Auswahl-Bedingungen für Variablen des Experiments und passende logische Verknüpfungen gelöst. Man kann zum Beispiel verlangen, daß:

(mindestens eine Spur zum Vertex assoziiert werden kann `ODER` mindestens 3 GeV entfernt vom Strahlrohr deponiert wurden) `UND` daß eine maximale Energiedeposition im rückwärtigen Kalorimeter nicht überschritten wird.

Ein neuer Ansatz ist der Einsatz neuronaler Netze. Dort werden Schnitte in einem vieldimensionalen Raum vorgenommen, der durch passend gewählte Variablen des Experiments aufgespannt wird.

Der Vorteil beim Einsatz neuronaler Netze ist einerseits ihre potentiell sehr kurze Zeit zur Entscheidungsfindung, andererseits können sie auch bisher ungenutzte Zusammenhänge erkennen und nutzen.

Verschiedene Experimente setzen bereits Neuronale Netze im Trigger ein. Das H1-Experiment setzt zum Beispiel in ihrem Level-2 Trigger neuronale Netze ein (siehe z.B. [1], [2]). Dieser Trigger ist modular aufgebaut, so daß für jede Physikklasse ein neuronales Netz eingesetzt werden kann.

Auch in der Analyse bereits gespeicherter Ereignisse treten häufig Probleme der Mustererkennung auf, insbesondere wenn die einzelnen Detektorinformationen zu übergeordneten Strukturen zusammengefügt werden. Bei LEP wurden erfolgreich neuronale Netze eingesetzt, um Jets zu erkennen, die aus einem b-Quark hervorgehen [3].

Bei ZEUS wird ein neuronales Netz eingesetzt, um gestreute Elektronen im Detektor zu identifizieren ([4]).

In dieser Arbeit wird das Problem der Erkennung von Jets mit neuronalen Netzen behandelt.

Die Erkennung von Jets ist ein notwendiger Schritt, um Rückschlüsse auf den eigentlichen Streuprozeß zu gestatten, der auf gestreute Quarks und Gluonen führt. Die Kenntnis des Zustandes der Quarks und Gluonen direkt nach einem Streuprozeß ist aber der Schlüssel zu vielen physikalischen Fragestellungen.

In dieser Arbeit soll untersucht werden, ob man unter Verwendung globaler Ereignisdaten und eines neuronalen Netzes die Anzahl der Jets in einem Ereignis bestimmen kann, ohne einen langwierigen Jet-Suchalgorithmus einzusetzen.

Mit dieser Arbeit soll ferner ein systematischer Zugang zur Suche nach geeigneten Eingabevariablen für ein neuronales Netz in Verbindung mit der Fouriertransformation eingesetzt werden.

2 Neuronale Netze

Da neuronale Netze im Mittelpunkt dieser Arbeit stehen, soll einleitend die Funktionsweise neuronaler Netze in Theorie und Anwendung dargestellt werden. Der Schwerpunkt liegt dabei auf den sog. Perceptrons, da hier ausschließlich Netze dieser Architektur verwendet werden.

2.1 Perceptrons

Perceptrons zählen zu den ersten neuronalen Netzwerken, die einer systematischen Untersuchung unterzogen wurden. In diesem Abschnitt soll auf diese wichtigen Netzwerke eingegangen werden.

Perceptrons sind reine feed-forward Netzwerke, die Information reist also während ihrer Verarbeitung nur in einer Richtung: von den Eingabeknoten x_i zu den Ausgabeknoten O_i. Charakteristisch für Perceptrons ist weiterhin, daß die Knoten in Schichten angeordnet sind. Man spricht von einer **Eingabeschicht**, einer **Ausgabeschicht** und eventuell von einer oder mehreren **verborgenen Schichten**. Zwei Perceptrons sind in Abb. 1 dargestellt. In (a) ist ein sogennantes **einfaches Perceptron** abgebildet (es besitzt keine versteckte Schicht); (b) zeigt ein Perceptron im allgemeinen Sinne, hier mit einer verborgenen Schicht. Die Nomenklatur folgt hier [5], in [6] ist das einfache Perceptron als ein einziger Knoten (oder auch Zelle) definiert.

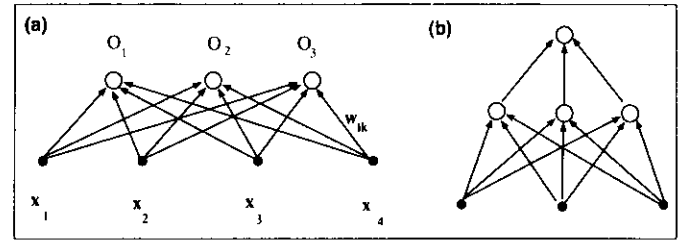


Abbildung 1: (a) einfaches Perceptron, (b) allgemeines Perceptron mit einer Schicht verborgener Knoten. x_i bezeichnet die Eingabeknoten, O_j die Ausgabeknoten

2.2 Funktionsweise des einfachen Perceptrons

Die Knoten eines neuronalen Netzes sind den Nervenzellen des menschlichen Körpers nachempfunden, wenn auch unter starken Vereinfachungen. Um zu entscheiden, ob ein Knoten ein Signal von sich gibt, wird im einfachsten Fall über alle Eingänge gewichtet summiert und die Summe dann mit einem Schwellenwert verglichen. Überschreitet die Summe den Schwellenwert, so „feuert" der Knoten (sein Ausgang liefert eine „1"), ansonsten bleibt der Knoten inaktiv (am Ausgang liegt eine „0").

Im Falle tatsächlich benutzter Netzwerke, wie bei ZEUS, werden statt dieser „binären" Knoten solche mit kontinuierlichen Ausgabewerten verwendet. In der Regel wird eine sigmoide Funktion der gewichteten Summe der Eingänge verwendet. Der Wertebereich einer solchen Ausgangsfunktion liegt typischerweise zwischen 0 und +1, bzw zwischen −1 und +1. Eine häufig verwendete Funktion ist die $g(x) = \tanh(x)$-Funktion, die einen Wertebereich zwischen −1 und +1 abdeckt. Abb. 2 zeigt ein (willkürliches) Beispiel einer sigmoiden Funktion.

Die Berechnung der Ausgabewerte eines Netzes erfolgt gemäß Gleichung 1:

$$O_i = g(h_i) = g(\sum_{k=1}^{K} w_{ik} x_k - \theta_i) \tag{1}$$

Die Formel gilt in dieser Form nur für das einfache Perceptron, kann aber leicht durch wiederholte Anwendung auf das allgemeine Perceptron übertra-

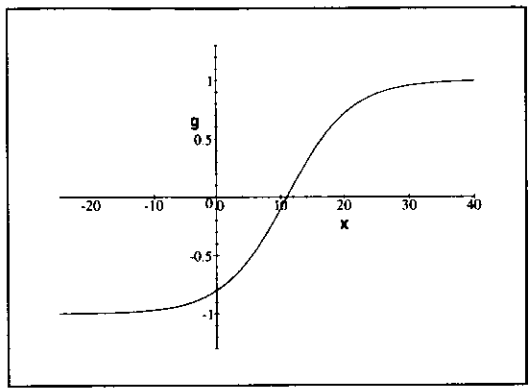


Abbildung 2: Die sigmoide Fuktion $\tanh(0.1 * x - 1.1)$

gen werden. Die Bezeichnungen sind größtenteils in Abb. 1 eingetragen: O_i ist der $i-$te Ausgabeknoten, die x_k bezeichnen die K Eingabeknoten, w_{ik} ist das Gewicht von Eingabeknoten k zu Ausgabeknoten i und θ_i ist der Schwellenwert von Ausgabeknoten i. $g(x)$ ist die Ausgabefunktion und hat häufig die Form $g(x) = \tanh(x)$

Sehr oft wird die explizite Nennung des Schwellenwertes in Gleichung 1 durch einen sogenannten **bias** ersetzt: Die Gleichung erhält dann folgende Form:

$$O_i = g(h_i) = g(\sum_{k=0}^{K} w_{ik} x_k) \tag{2}$$

Hierbei wird der Schwellenwert wie eine Verbindung zu einem konstanten Eingang -1 mit Gewicht θ_i behandelt, also $x_0 = -1$ und $w_{i0} = \theta_i$ (gemäß [5]).

Die Aufgabe eines neuronalen Netzes soll es sein, für jeden Satz von Eingabewerten, der den Eingangsknoten präsentiert wird, Ausgabewerte zu berechnen, die möglichst nahe an einem vorgegebenen Satz von Zielwerten liegen. Im optimalen Fall reproduziert das neuronale Netz diese Werte exakt,

im Regelfall wird es diese nur approximieren.

Verwendet man die Bezeichnungen aus Abb. 1 und erweitert man diese um einen Index μ, der über die M Ein- und Ausgabemuster läuft, so werden die Eingabemuster auch als **Eingabevektor** x_i^μ, die Ausgabewerte als **Ausgabevektor** O_i^μ bezeichnet.

Das Netz löst damit Klassifikationsprobleme, die sich bei Verwendung von nur zwei Variablen durch eine graphische Darstellung in zwei Dimensionen noch anschaulich interpretieren lassen. Die Funktionsweise eines Knotens mit der Stufenfunktion

$$g(x) = \begin{cases} 1 & , x \geq 0 \\ -1 & , x < 0 \end{cases} \tag{3}$$

als Ausgabefunktion entspricht dabei dem Aufteilen des Raumes der Eingabewerte mittels einer Geraden.

Bei N Eingabeknoten spannen die Eingabevektoren $\vec{x} = (x_1, x_2, \ldots, x_N)$ einen N-dimensionalen Raum auf, in dem eine derartige Ausgabefunktion den Raum der Eingabewerte durch eine Hyperebene trennt. Die Bedingung

$$\vec{w_i} \cdot \vec{x} > 0,$$

die in diesem Falle für das „Feuern" des Knotens erfüllt werden muß, entspricht gerade dem Aufteilen des Eingaberaumes durch eine (N-1)-dimensionale Hyperebene, die senkrecht auf dem Vektor der Gewichte $\vec{w_i} = (w_{i1}, w_{i2}, \ldots, w_{iN})$ steht. (Dabei ist der Schwellenwert $\theta_i = 0$ gesetzt.)

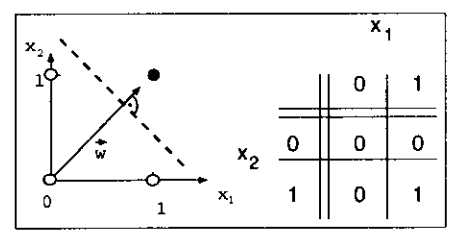


Abbildung 3: links: Aufteilung des Eingaberaumes durch eine Hyperebene senkrecht auf dem Gewichtsvektor $\vec{w}$, rechts: Wahrheitstabelle der logischen UND-Funktion

Abb. 3 demonstriert dies für den 2-dimensionalen Fall an einem simplen Beispiel, der logischen UND-Funktion. Die Hyperebene ist definiert durch $\vec{w}_i$, auf dem sie senkrecht steht, und durch den Schwellenwert θ_i, der den senkrechten Abstand der Ebene zum Ursprung angibt. Alle Punkte auf der Seite des Ursprungs, d.h. $(0,0)$, $(1,0)$ und $(1,0)$ haben einen Ausgabewert von 0; der Punkt $(1,1)$ liegt auf der anderen Seite der Hyperebene (in 2 Dimensionen eine Gerade) und hat den Ausgabewert 1.

2.3 Trainieren eines Netzes

Im Gegensatz zu herkömmlichen Algorithmen wird die Leistung eines neuronalen Netzes nicht durch ein starres Programm festgelegt, es durchläuft vielmehr einen **Trainings-** oder auch **Lernprozeß**. Dabei werden die Parameter des Netzes, also die Schwellenwerte und Gewichte, entsprechend der Abweichung der Ausgabewerte des Netzes von Referenzwerten modifiziert.

Der Trainingsprozeß stellt neben der Wahl der Topologie des Netzes die größte Herausforderung beim Einsatz neuronaler Netze dar, da es nur bei den einfachsten Netzen möglich ist, die Gewichte und Schwellenwerte „von Hand" festzulegen. Bei komplexeren Anwendungen muß ein neuronales Netz erst „trainiert" werden. Es wird dabei zwischen zwei Lernprinzipien unterschieden: **beaufsichtigtes Lernen** und **unbeaufsichtigtes Lernen**.

Bei unbeaufsichtigtem Lernen werden dem Netz wiederholt Eingabemuster präsentiert, in denen das Netz selbständig nach einem Muster suchen muß.

Bei beaufsichtigtem Lernen wird dem Netz zu jedem Trainingsmuster auch die erwünschte Lösung präsentiert. Abhängig von der Trainingsmethode werden dem Netz ein oder mehrere Muster präsentiert, bevor die Gewichte und Schwellenwerte so modifiziert werden, daß die Ausgabe näher an dem Referenzwert liegt. Als Maß für diese 'Nähe' wird in den meisten Fällen eine **Fehlerfunktion E** verwendet. Eine gängige Definition lautet (unter Verwendung von Gl. 2):

$$\begin{aligned} E(\vec{w}_i, \theta_i) &= \frac{1}{2}\sum_{i\mu}(\zeta_i^\mu - O_i^\mu)^2 \\ &= \frac{1}{2}\sum_{i\mu}\left[\zeta_i^\mu - g\left(\sum_k w_{ik}x_k^\mu\right)\right]^2 \end{aligned} \tag{4}$$

Dabei sind O_i^μ die Ausgabewerte des Netzes und ζ_i^μ die Referenzwerte, die dem Netz präsentiert werden; $g(x)$ ist die Ausgangsfunktion der Knoten, x_k^μ ist der Vektor der K Eingabewerte für Muster μ. Der Index i läuft über die Ausgangsknoten, der Index k über die Eingangsknoten und der Index μ über alle Eingabemuster.

Die Fehlerfunktion (auch **Energiefunktion**) E ist abhängig von den Gewichten und Schwellenwerten der Knoten und erreicht ein globales Minimum, wenn diese optimal gewählt sind.

In der Regel ist es nicht möglich, die Fehlerfunktion analytisch zu minimieren, so daß ein iterativer Prozeß eingesetzt werden muß. Hierzu gibt es verschiedene Methoden, siehe zum Beispiel [7]. Eine **Iteration** im Sinne des Trainings neuronaler Netze endet dabei jeweils mit einer Modifikation der Netzwerkparameter.

Eine Idee dabei ist es, die Fehlerfunktion „herabzurutschen", bis das Minimum erreicht ist. Die Bezeichnung für dieses Verfahren ist **„gradient descent"**-Algorithmus. Der Name entstammt der Eigenart des Verfahrens, die Gewichte so zu verändern, daß man sich abwärts auf der Fehlerfunktion bewegt, gerade entgegen der Richtung des Gradienten der Fehlerfunktion in Bezug auf die Gewichte.

Die Gewichte w_{ik} und Schwellenwerte θ_i werden um einen Wert Δw_{ik} ($\Delta\theta_i$) verändert. Unter Verwendung von Gl. 4 und der Abkürzung $h_i^\mu = \sum_k w_{ik}x_k^\mu$ stellt sich das Verfahren so dar:

$$\begin{aligned} \Delta w_{ik} &= -\eta\frac{\partial E}{\partial w_{ik}} \\ &= \eta\sum_\mu(\zeta_i^\mu - O_i^\mu)g'(h_i^\mu)x_k^\mu \end{aligned} \tag{5}$$

$$\begin{aligned} \text{und } \Delta\theta_i &= -\eta\frac{\partial E}{\partial \theta_i} \\ &= -\eta\sum_\mu(\zeta_i^\mu - O_i^\mu)g'(h_i^\mu) \end{aligned} \tag{6}$$

Dabei bezeichnet man η als den **Lernparameter**, der wesentlich für das Konvergenzverhalten ist: Wählt man ihn zu groß, kann man das Minimum verfehlen, wählt man η zu klein, konvergiert das Verfahren nur sehr langsam.

Vor der Berechnung der Modifikationen der Gewichte (Gleichungen 5 und 6) wird hier die Fehlerfunktion über alle Muster summiert. Dadurch soll verhindert werden, daß einzelne Muster einen unverhältnismäßig großen Einfluß

auf die Änderung der Parameter haben. Es ist sogar denkbar, daß ein Muster zu den Trainingsmustern hinzugefügt wurde, obwohl es bei näherem Betrachten aussortiert würde, etwa weil es aus fehlerhaften Berechnungen stammt oder mit einem falschen Referenzwert versehen wurde. Dieses Muster könnte nun für sich gesehen Änderungen der Gewichte verursachen, die in eine unerwünschte Richtung führen und zudem deutlich größer sind als die der als repräsentativ angesehenen Muster. Wird vor der Modifikation der Gewichte die Fehlerfunktion über mehrere Muster gemittelt, so werden die unerwünschten Modifikationen aufgrund des Ausreißers mit der Anzahl der Muster unterdrückt.

Werden die Änderungen dagegen nach jedem Muster getätigt, so entfällt die Summe über μ:

$$\Delta w_{ik} = \eta(\zeta_i^\mu - O_i^\mu)g'(h_i^\mu)x_k^\mu = \eta\delta_i^\mu x_k^\mu \tag{7}$$

mit $\delta_i^\mu \equiv (\zeta_i^\mu - O_i^\mu)g'(h_i^\mu)$. Gleichung 7 erhält ihren Namen von dem δ: Sie wird gemeinhein als **delta rule** bezeichnet. **Adaline rule**, **Widrow-Hoff rule** und **LMS rule** (least mean squares) sind auch gebräuchlich [5].

2.4 Mehrschichtige Netze

Einfache Perceptrons habe natürliche Beschränkungen, die schon bei Verwendung einer Schicht verborgener Knoten entfallen. So zum Beispiel kann die logische Funktion XOR nicht durch ein einfaches Perceptron dargestellt werden, wohl aber mit Hilfe einer verborgenen Schicht [5].

An dieser Stelle soll daher der Formalismus auf neuronale Netze mit einer verborgenen Schicht erweitert werden. Die im Text verwendeten Bezeichnungen sind zur Orientierung in Abb. 4 eingetragen. Es bezeichnen x_k, $k = 1..K$ die Eingabewerte, w_{jk} die Eingangsgewichte der J verborgenen Knoten, α_j deren Schwellenwerte und V_j deren Ausgangswerte ($j = 1..J$). Analog bezeichnen W_{ij} die Gewichte, β_i die Schwellenwerte und O_i die Ausgangswerte der I Ausgabeknoten ($i = 1..I$).

Obwohl man für einfache Perceptrons Lernmethoden anwenden kann, deren Konvergenz im Falle linear separierbarer Probleme mathematisch bewiesen ist[5], gilt dies nicht für mehrschichtige Perceptrons. Für diese gibt es zwar mehrere Lernalgorithmen, es kann jedoch in der Regel die Konvergenz nicht bewiesen werden, so daß man häufig mehrere Netzwerke trainieren und deren Leistung vergleichen muß.

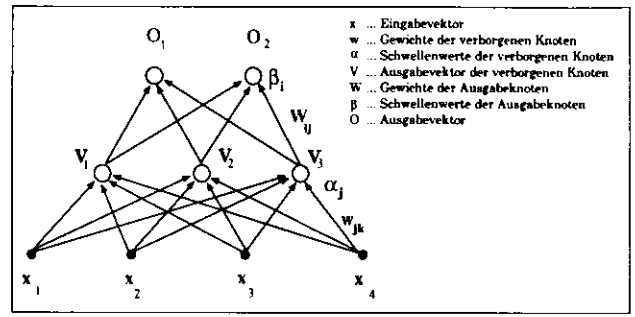


Abbildung 4: Mehrschichtiges neuronales Netz

Ein Lernalgorithmus wird im Folgenden detailliert vorgestellt: der **back-propagation** Algorithmus.

2.4.1 Back-propagation

Entsprechend Abb. 4, ist die Ausgabe des versteckten Knotens j:

$$V_j^\mu = g(h_j^\mu) = g(\sum_{k=1}^{K} w_{jk}x_k^\mu - \alpha_j)$$

μ ist wieder der Index des Musters. Ausgabeeinheit i gibt demnach aus:

$$O_i^\mu = g(h_i^\mu) = g\left(\sum_{j=1}^{J} W_{ij}V_j^\mu - \beta_i\right) = g\left(\sum_{j=1}^{J} W_{ij} \cdot g\left(\sum_{k=1}^{K} w_{jk}x_k^\mu - \alpha_j\right) - \beta_i\right)$$

Mit obiger Definition wird die Fehlerfunktion

$$\begin{aligned} E &= \frac{1}{2}\sum_{\mu i}[\zeta_i^\mu - O_i^\mu]^2 \\ &= \frac{1}{2}\sum_{\mu i}\left[\zeta_i^\mu - g\left(\sum_j W_{ij}g\left(\sum_k w_{jk}x_k^\mu - \alpha_j\right) - \beta_i\right)\right]^2 \end{aligned} \tag{8}$$

wobei wiederum ζ_i^μ die Referenzmuster sind.

Der „gradient descent" Algorithmus kann angewandt werden, wenn g eine differenzierbare Funktion der Gewichte und Schwellenwerte w_{jk}, W_{ij}, α_j und β_i ist. Für die Änderung jeder Größe gilt nach [8]:

$$y^{(\nu+1)} = y^{(\nu)} - \eta \cdot \Delta y^{(\nu)} \tag{9}$$

$$\Delta y^{(\nu)} = \frac{\partial E}{\partial y} + \kappa \cdot \Delta y^{(\nu-1)} \tag{10}$$

wobei y für die zu modifizierende Größe $(w_{jk}, W_{ij}, \alpha_j, \beta_i)$ steht und ν die Nummer der Trainingsiteration bezeichnet.

Dieses Gleichungspaar entspricht Gl. 5 mit einem Zusatzterm $\kappa\Delta y^{(\nu-1)}$. Dieser Term stellt eine Modifikation des simplen back-propagation Algorithmus dar und wird gemeinhin als **momentum Term** oder auch Impulsterm bezeichnet. Auf diese Weise wird die nächste $(\nu+1)$ Änderung die Tendenz haben, der vorigen $(\nu-1)$ zu folgen. Auf diese Weise sollen Oszillationen des Verfahrens vermieden werden.

Betrachtet man nun die partiellen Ableitungen, so ergibt sich für die Gewichte und Schwellenwerte der Ausgangsknoten:

$$\begin{aligned}\frac{\partial E}{\partial W_{ij}} &= -\sum_\mu [\zeta_i^\mu - g(\sum_j W_{ij} V_j^\mu - \beta_i)] g'(\sum_j W_{ij} V_j^\mu - \beta_i) V_j^\mu \\ &= -\sum_\mu [\zeta_i^\mu - O_i^\mu] g'(h_i^\mu) V_j^\mu \equiv -\sum_\mu \delta_i^\mu V_j^\mu \end{aligned} \tag{11}$$

$$\frac{\partial E}{\partial \beta_i} = \sum_\mu [\zeta_i^\mu - O_i^\mu] g'(h_i^\mu) = \sum_\mu \delta_i^\mu \tag{12}$$

Dabei gilt $\delta_i^\mu \equiv [\zeta_i^\mu - O_i^\mu] g'(h_i^\mu)$.

Für die verborgenen Knoten gilt:

$$\begin{aligned}\frac{\partial E}{\partial w_{jk}} &= \frac{\partial E}{\partial V_j^\mu} \frac{\partial V_j^\mu}{\partial w_{jk}} \\ &= -\sum_{i,\mu} [\zeta_i^\mu - O_i^\mu] g'(h_i^\mu) W_{ji} g'(h_j^k) x_k^\mu \\ &= -\sum_{i,\mu} \delta_i^\mu W_{ij} g'(h_j^k) x_k^\mu \\ &= -\sum_\mu \delta_j'^\mu x_k^\mu \end{aligned} \tag{13}$$

$$\begin{aligned}\frac{\partial E}{\partial \alpha_j} &= \sum_{i,\mu} \delta_i^\mu W_{ij} g'(h_j^k) \\ &= \sum_\mu \delta_j'^\mu \end{aligned} \tag{14}$$

Dabei wird gesetzt: $\delta_j'^\mu \equiv g'(h_j^\mu) \sum_i W_{ij} \delta_i^\mu$.

Wie man den Gleichungen 11–14 entnehmen kann, werden die Änderungen der verborgenen Knoten aus den gewichteten Summen h_i^μ der Ausgangsknoten berechnet. Der Informationsfluß ist also beim Training des Netzes von „vorne" nach „hinten", woher dieser Algorithmus auch den Namen **back-propagation** erhält.

2.4.2 Modifizierter back-propagation Algorithmus

Während des Anfertigens dieser Arbeit wurde ein Programm benutzt, das eine Abwandlung des back-propagation Algorithmus mit Impulsterm benutzt [9].

Wesentlich bei diesem Verfahren ist die Einführung einer **Temperatur**, deren hauptsächliche Aufgabe es ist, lokale Minima der Energiefunktion zu unterdrücken.

Bei diesem Verfahren werden zusätzlich die Quadrate der Gewichte auf 1 normiert, so daß der Einfluß der Initialisierung vermindert wird. Die Matrizen w_{jk} und W_{ij} werden Reihenweise normiert, so daß folgende Bedingungen erfüllt sind:

$$\sum_{j=1}^{J} W_{ij}^2 = 1, \forall i$$

$$\sum_{k=1}^{K} w_{jk}^2 = 1, \forall j$$

Diese Normierung kann in der Implementation (dEXTRa, siehe Anhang) wahlweise an- oder abgestellt werden. Ist sie angewählt, wird die Normierung nach jeder Trainingsiteration durchgeführt.

Das η in Gleichung 9 ist jetzt eine Funktion der Temperatur:

$$\eta = 1 + \gamma - tanh^2 \frac{1}{t}$$

Der Wert γ stellt sicher, daß η auch bei sehr kleinen Temperaturen spürbar größer als 1 bleibt und stellt somit sicher, daß die Konvergenz nicht zu stark verlangsamt wird für $t \to 0$.

Die Temperatur ist in diesem Verfahren eine Variable und konvergiert im Falle eines durch das zu trainierende Netzwerk lösbaren Problems gegen 0:

$$t^{(\nu+1)} = t^{(\nu)} - \frac{\eta}{10}\Delta t^{(\nu)}$$
$$\Delta t^{(\nu)} = \frac{\partial E^{(\nu)}}{\partial t} + \kappa \Delta t^{(\nu-1)}$$

Die Übertragungsfunktion lautet jetzt:

$$g(x) = \frac{1}{2}\left(1 + tanh\left(\frac{x}{t}\right)\right)$$

Dieser Algorithmus ist in dem Programm **dEXTRa** implementiert. Es handelt sich dabei um ein Fortran-Programm, das sowohl ein neuronales Netz trainieren als auch einsetzen kann. Die gewünschte Funktionalität wird durch eine Konfigurationsdatei bestimmt, die im Anhang näher beschrieben ist.

In dieser Datei legt man unter anderem die Parameter des Netzes fest, typischerweise sind die Ausgangswerte ([9]):

Parameter	Bedeutung des Parameters	(anfänglicher) Wert
T	Ausgangstemperatur	5.0
κ	Gewicht des Impulsterms	ungefähr 0.1
η	Lernparameter	ungefähr $\eta \leq 0.001$, aber auch bis zu $\eta \approx 0.1$.
γ	Mindeständerung von η	ungefähr ≈ 0.1
N	Anzahl der Muster, über die bei der Berechnung von E summiert werden soll	meist gleich der Anzahl der Trainingsmuster

Diese Werte können allerdings nur als Richtwerte angesehen werden, bei jeder konkreten Anwendung gilt es, einen optimalen Satz an Parametern zu suchen.

2.5 Ein Beispiel: Silbentrennung mittels NN

Als ein Anwendungsbeispiel für neuronale Netze wurde ein Netz zur Silbentrennung trainiert.

Es handelt sich dabei um ein Perceptron mit einer verborgenen Schicht, gemäß Abb. 5

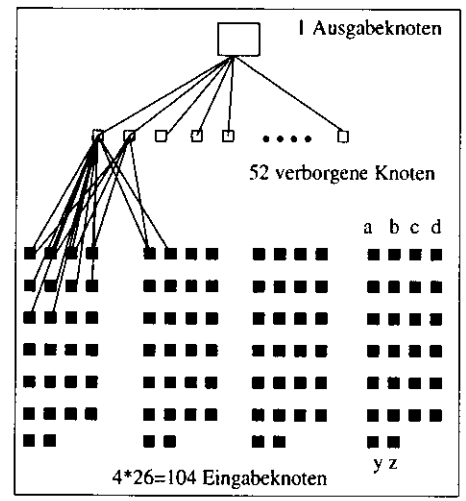


Abbildung 5: Neuronales Netz zur Silbentrennung: Perceptron mit einer verborgenen Schicht.

Das Netz besitzt $4*26 = 104$ Eingabeknoten mit denen 2 Buchstaben vor und 2 nach einer möglichen Trennstelle als Eingabe benutzt werden können.

Die ersten 26 Eingabeknoten entsprechen dem Buchstaben 2 Positionen vor einer möglichen Trennstelle, die nächsten 26 entsprechen dem Buchstaben direkt vor der möglich Trennstelle, dementsprechend gehören die nächsten 52 Knoten den Buchstaben nach der möglichen Trennstelle.

Der jeweils aktive Knoten innerhalb einer 26er-Blocks ergibt sich aus dem Alphabet: Soll ein 'A' dargestellt werden, so ist der 1. Knoten auf '1' gesetzt, alle anderen auf '0'. Für ein 'C' ist dementsprechen nur der 3. Knoten auf '1'

gesetzt, usw. Das Leerzeichen wird durch 26 Nullen ausgedrückt.

Denkbar wäre auch, den Eingabeknoten reelle Zahlen als Eingabe zu bieten und die Buchstaben als reelle Intervalle zu codieren. Damit könnte man zwar die Anzahl der Eingabeknoten reduzieren, allerdings würde man dadurch verschiedenen Buchstaben a priori eine unterschiedliche Gewichtung zukommen lassen.

Die versteckte Schicht umfaßt 52 Knoten, die Ausgabe erfolgt durch einen einzelnen Knoten.

2.5.1 Training

Zum Training des Netzes wurden als Trainingsdatensatz 200 willkürlich gewählte englische Worte mittels des Trennalgorithmus von TEX mit Trennstrichen versehen und von einem zu diesem Zweck erstellten Programm in Trainingsmuster für das neuronale Netz konvertiert. Dabei wurde ein 4-Buchstaben-Fenster über das Wort geschoben, an den Wortenden wurde jeweils ein Leerzeichen eingefügt. Befindet sich eine Trennstelle in der Mitte des Fensters, so sollte das Netz '1' ausgeben, ansonsten '0'. In Abbildung 6 sind zwei Trainingsmuster für das Netz schematisch abgebildet.

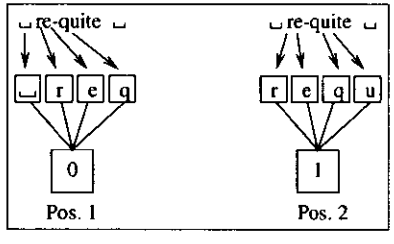


Abbildung 6: Trainingsmuster für das neuronale Netz zur Wörtertrennung. Pos.1 zeigt ein Muster für 'keine Trennung', Pos.2 zeigt ein Muster für 'Trennung'

Das Training wurde begonnen mit den Anfangswerten:

$$t = 5.0$$

$$\eta = 0.05$$
$$\kappa = 0.005$$

Im Laufe einer jeden Iteration des Trainings wurden dem neuronalen Netz neun Muster des Trainingsdatensatzes präsentiert, bevor die Modifikationen der Gewichte vorgenommen wurden.

Bereits nach ca. 40 Iterationen hatte das Netz einen nahezu minimalen Fehler erreicht, die Überprüfung des Lernerfolges fand mit einem Testdatensatz von 102 Wörtern statt, die unabhängig von den ersten 200 gewählt wurden. In Abbildung 7 ist der Trainingsverlauf festgehalten:

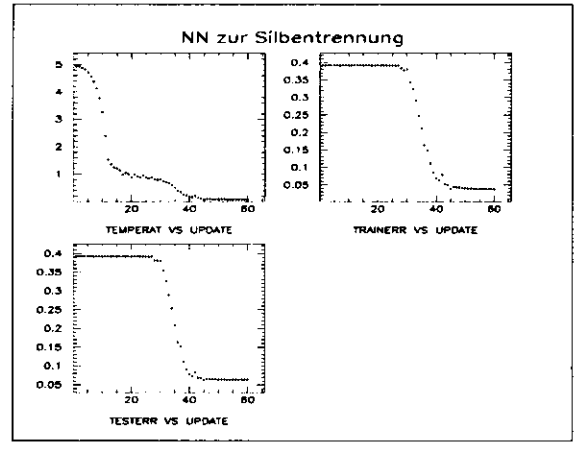


Abbildung 7: Trainingsverlauf des NN zur Wörtertrennung. Als Abszisse dient stets die Nummer der Trainingsiteration, als Ordinate sind dargestellt: die Temperatur (TEMPERAT) des Netzes und die Fehler für den Trainings- (TRAINERR) und den Testdatensatz (TESTERR). Für die Berechnung der Fehler siehe Gl. 15

Aufgetragen sind die Temperatur (TEMPERAT) und die Fehler für die Trainingsmuster (TRAINERR) und für die Testmuster (TESTERR), als Abszisse dient die Nummer der Trainingsiteration (UPDATE). Die Testmuster

dienen als Indikator für **Übertraining**, d.h. ob das Netz die Trainingsbeispiele 'auswendig gelernt' hat, statt eine allgemeine Lösung für das Problem zu finden. Bei Anwesenheit von Übertraining hätte der Fehler der Trainingsmuster zwar weiterhin abgenommen, der Fehler der Testmuster wäre jedoch in die Höhe geschnellt.

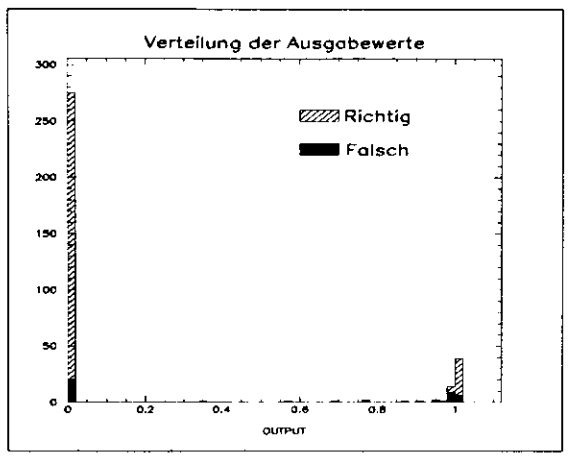


Abbildung 8: Verteilung der richtigen/falschen Ausgaben des NN. Richtig bedeutet, daß der gemäß Gleichung 16 gerundete Ausgabewert OUTPUT des Netzes mit der Referenzantwort übereinstimmt, andernfalls liegt eine falsche Antwort vor.

Der Fehler berechnet sich zu:

$$\left.\begin{array}{l}\text{TRAINERR}\\ \text{TESTERR}\end{array}\right\} = \frac{1}{2N}\sum_{\mu=1}^{N}(\Theta(O^\mu) - \zeta^\mu)^2 \qquad (15)$$

Dabei ist N die Anzahl der Trainings- bzw. Testmuster, μ läuft über alle Muster $\mu = 1..N$, ζ^μ ist die Referenzantwort für Muster μ, O^μ ist der Ausgabewert des Netzes und die Funktion $\Theta(O^\mu)$ ist definiert als:

$$\Theta(O^\mu) = \begin{cases} +1, & O^\mu > 0.5 \\ 0, & O^\mu \leq 0.5 \end{cases} \qquad (16)$$

Die Verteilung der richtigen und falschen Antworten des neuronalen Netzes sind in Abb. 8 aufgetragen. Eine richtige Antwort des Netzes liegt vor, wenn der gemäß Gleichung 16 gerundete Ausgabewert des Netzes mit der Referenzantwort übereinstimmt. Stimmen beide nicht überein, liegt eine falsche Antwort vor. OUTPUT bezeichnet den Ausgabewert des Netzes.

2.5.2 Ergebnis

Obwohl das Netz nur mit 200 Wörtern trainiert wurde, setzt es bei 102 völlig unbekannten Wörtern die Trennstriche in 76% der Fälle richtig, wobei es auch einen „falschen" Trennstrich für je ca drei von TEX gesetzten vorschlägt. „Falsch" bedeutet, daß TEX diesen nicht vorgeschlagen hatte.

Zur Auswertung wurde ein Ausgabewert ≤ 0.5 als 'keine Trennung', ein Wert > 0.5 als 'Trennung' interpretiert.

Tabelle 1 zeigt Beispiele für die Ausgabe des Netzes, zusammen mit den entsprechenden Vorschlägen von TEX. Die Beispiele wurden willkürlich aus dem Testdatensatz entnommen.

2.5.3 Verkleinerung des Netzes

Als abschließender Test wurde das Netz mit einer geringeren Anzahl verborgener Knoten trainiert. Diese wurde bis auf nur 3 Knoten reduziert, wobei nach wie vor ca. 80% der von TEX gesetzten Trennstriche richtig reproduziert wurden, allerdings nahm das Verhältnis der falsch gesetzten zu den richtigen von ca. 1:3 auf ca 2:3 zu. Außerdem wurden die Entscheidungen undeutlicher. Abb. 9 zeigt die Ausgabe von Netzwerken mit 35, 15, 8 und 3 verborgenen Knoten (v.l.o.n.r.u.).

Bemerkenswert ist, daß erst zwischen 8 und 3 verborgenen Knoten die Fehler spürbar zunehmen.

2.5.4 Verbesserungsmöglichkeiten

Sollte ein neuronales Netz wie dieses tatsächlich zur Silbentrennung eingesetzt werden, so müßte es noch verbessert werden. Einige Möglichkeiten sind:

- Vergrößerung des Trainingssamples. Es wurde mit nur 200 Wörtern trainiert, die kaum alle Regeln der Kunst des Silbentrennens umfassen können.

Vorschlag von TEX	Vorschlag des Netzes
con-tin-gent	con-tin-gent
dis-tend	distend
pre-rog-a-tive	prero-ga-tive
pro-mul-gate	pro-m-ul-gate
plain-tiff	plain-tiff
abom-inable	abomi-nable
vi-ti-ate	vitiate
gam-bol	gam-bol
in-di-gent	in-digent
equiv-o-cal	equi-vo-cal
unc-tu-ous	unctu-ous
gar-ri-son	gar-ri-son
tru-ant	truant
tid-ings	tidings
enun-ci-ate	enun-ciate
pul-pit	pul-pit
fac-ti-tious	fac-titious
im-bue	im-bue
con-ju-ga-tion	con-ju-ga-tion
scur-rilous	scur-ri-lous
this-tle	this-tle
ef-fi-ca-cious	ef-fi-ca-cious
prim-rose	pri-mrose
ex-pos-tu-la-tion	ex-pos-tu-lation
cu-rate	curate
jab-ber	jabber
in-ter-locu-tor	in-ter-locu-tor

Tabelle 1: Ausgabe des Neuronalen Netzes zur Silbentrennung

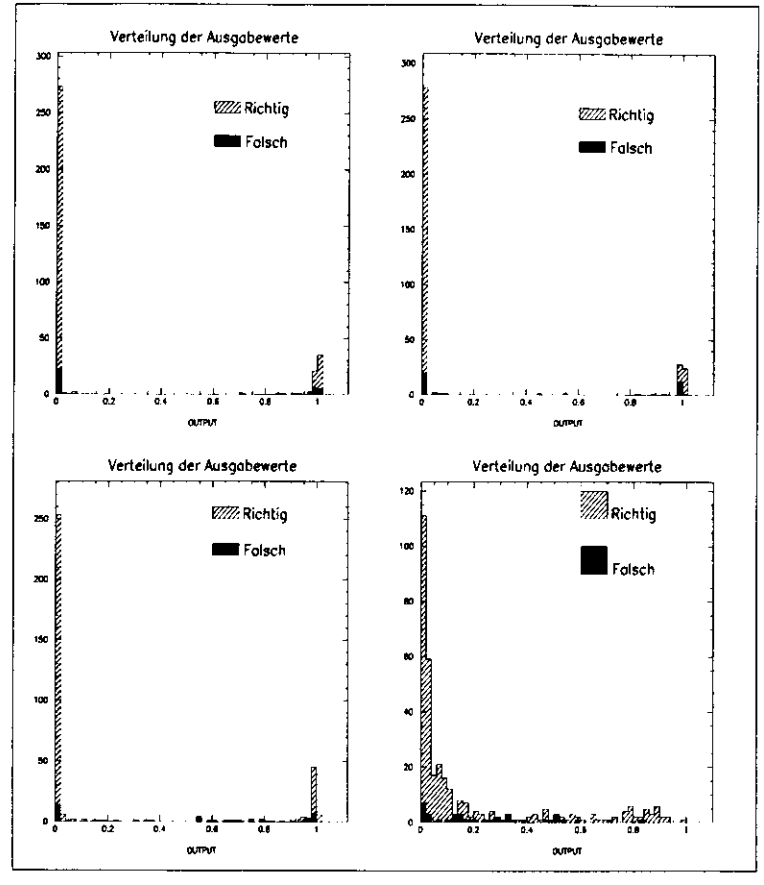


Abbildung 9: Verteilung der richtigen/falschen Ausgaben des NN. V.l.o.n.r.u.: 35, 15, 8 und 3 Knoten in der verborgenen Schicht. Richtig bedeutet, daß der gemäß Gleichung 16 gerundete Ausgabewert OUTPUT des Netzes mit der Referenzantwort übereinstimmt, andernfalls liegt eine falsche Antwort vor.

- Vergrößerung des „Trennfensters". In dem verwendeten Netz wurden diesem nur 4 Buchstaben gezeigt, auf Grund derer eine Entscheidung gefällt werden mußte. Eine Erweiterung auf 6 oder 8 Buchstaben sollte daher die Leistungsfähigkeit verbessern.

- Das Leerzeichen hätte als ein weiteres Zeichen codiert werden sollen, so daß das Netzwerk Wortanfang und -ende besser hätte erkennen können.

3 Der ZEUS Detektor

Die Anwendung der neuronalen Netze in dieser Arbeit befaßt sich mit Daten des ZEUS-Detektors, wobei praktisch ausschließlich die Daten des Uran-Kalorimeters benutzt werden. Daher wird an dieser Stelle zunächst das Kalorimeter beschrieben.

Der ZEUS-Detektor ist ein fast hermetischer Universaldetektor [10], der in [11] und [12] ausführlich beschrieben ist.

Im folgenden wird der Aufbau des Kalorimeters skizziert, da in dieser Arbeit hauptsächlich Informationen dieser Komponente verwendet werden.

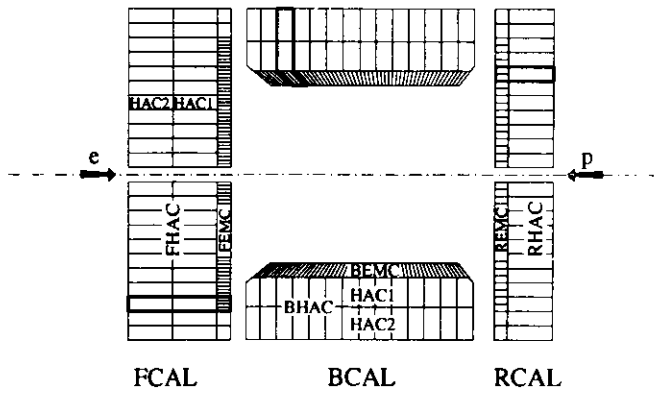


Abbildung 10: Aufbau des Kalorimeters

ZEUS verwendet ein kompensierendes Kalorimeter, das in drei Bereiche unterteilt ist (siehe Abb. 10): *FCAL, BCAL* und *RCAL*. Der vordere Bereich (FCAL) (in Flugrichtung der Protonen) deckt in Bezug auf den Wechselwirkungspunkt einen Polarwinkelbereich $2.6° < \vartheta < 36.7°$ ab. Der mittlere Bereich, der sich zylindrisch um das Strahlrohr legt, deckt den Bereich $36.7° < \vartheta < 129.1°$ ab, während der hintere Bereich (RCAL) Energiedepositionen im Winkelbereich $129.1° < \vartheta < 176.2°$ registriert.

Jeder Bereich des Kalorimeters ist in eine elektromagnetische (EMC) und eine (RCAL) bzw. zwei (FCAL + BCAL) hadronische Sektionen (HAC) unterteilt, dabei befindet sich die elektromagnetische Sektion vom Wechselwirkungspunkt gesehen vor der hadronischen. Die EMC-Sektionen sind nochmal in Zellen der Größe $5x20\text{cm}^2$ (FCAL,BCAL) bzw. $10x20\text{cm}^2$ (RCAL) unterteilt, die HAC-Sektionen sind durchgängig in $20x20\text{cm}^2$-Zellen unterteilt.

Die einzelnen Zellen des Detektors sind aus abwechselnden Uran- und Szintillatorschichten aufgebaut. Das in den Szintillatoren erzeugte Licht wird von zwei Photomultipliern je Zelle ausgelesen. Die Uranschichten dienen einerseits zu einer möglichst großen räumlichen Ausbildung von elektromagnetischen bzw. hadronischen Schauern schon bei geringer Schichtdicke, andererseits erlauben sie eine Eichung des Kalorimeters anhand der bekannten Radioaktivität.

Einzelne Zellen sind im Kalorimeter zu Türmen zusammengefaßt, für jeden Detektorbereich ist in Abb. 10 ein Turm dick umrandet. Im FCAL besteht ein Turm beispielsweise aus vier FEMC-, einer FHAC1- und einer FHAC2-Zelle.

Das Kalorimeter ist so konstruiert, daß die unterschiedlichen Schauereigentschaften von Hadronen und Elektronen/Photonen so kompensiert werden, daß bei gleicher Eingangsenergie zeitintegriert gleiche Signalhöhen erreicht werden. Man spricht daher von einem kompensierenden Kalorimeter. Dieser Effekt wird unter anderem durch entsprechend gewählte Verhältnisse der Dicken der Uran- und Szintillatorschichten der Kalorimeterzellen erreicht.

FCAL und RCAL sind in frontaler Ansicht in Abb. 11 dargestellt, man erkennt deutlich die höhere Granularität des FCALs.

Das Kalorimeter erreicht eine mittels Teststrahlmessungen bestimmte Energie-Auflösung von

$\sigma_E/E = 0.18/\sqrt{E(\text{GeV})}$ für Elektronen und

$\sigma_E/E = 0.35/\sqrt{E(\text{GeV})}$ für Hadronen.

Die absolute Energieskala von FCAL und BCAL ist zur Zeit bis auf 3%

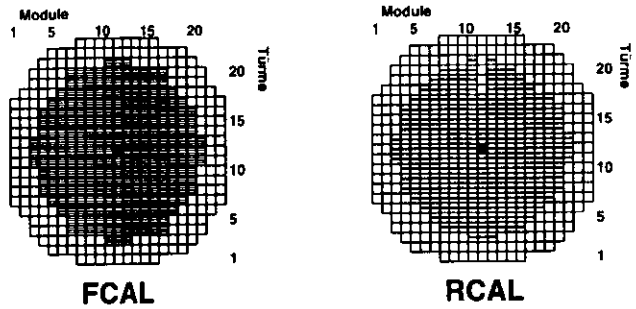


Abbildung 11: FCAL und RCAL senkrecht zur Strahlachse. Die Bereiche mit höherer Granularität gehören zum elektromagnetischen Kalorimeter EMC, der umliegende Bereich bildet das HAC0.

bekannt.

Die zeitliche Auflösung der Kalorimeterzellen ist bei Energiedepositionen über 4.5 GeV besser als 1 ns.

Die Signale des Kalorimeters müssen ein dreistufiges Triggersystem durchlaufen, bevor sie endgültig auf Magnetband geschrieben werden. Jede Detektorkomponente hat dabei eine eigene erste und zweite Triggerstufe, im folgenden wird nur auf das Triggersystem des Kalorimeters eingegangen. Auf jeder Triggerstufe werden die Informationen der komponenteneigenen Triggersysteme in einem globalen Trigger zusammengeführt.

In der ersten Triggerstufe (FLT) werden die Signale des Kalorimeters auf zusammenhängende Einträge abgesucht. Diese Signale müssen oberhalb von Schwellenwerten liegen, die von der Detektorregion und dem untersuchten Physikprozeß abhängig sind. In der zweiten Stufe (SLT) werden bereits kompliziertere Strukturen untersucht, es werden verschiedene Regionen des Kalorimeters miteinander verknüpft [13]. Im Anschluß an den SLT gelangen die Ereignisdaten zu dem „Ereignisbauer" (event builder), dessen Aufgabe es ist, sämtliche Daten eines Ereignisses zu sammeln und als eine Einheit weiterzugeben. In der anschließenden dritten Triggerstufe (TLT) wird das Ereignis rekonstruiert und mit zeitaufwendigeren Routinen untersucht, als es die ersten beiden Triggerstufen auf Grund der hohen Datenrate zugelassen hätten.

In den Daten des Kalorimeters sind stets Untergrundsignale aus mehreren Quellen enthalten, hierzu trägt sowohl das Uran bei als auch die Ausleseelektronik. Um störende Einflüsse dieser Untergrundsignale zu unterdrücken, werden nur Kalorimeterzellen betrachtet, deren Energieeinträge $60MeV$ in den EMC-Sektionen und $110MeV$ in den HAC-Sektionen überschreiten.

Eine Beschreibung der Rekonstruktionssoftware für das Kalorimeter findet sich in [14] und [15].

3.1 Das ZEUS-Koordinatensystem

Bei der Beschreibung des ZEUS-Detektors wird per Konvention ein rechtshändiges Koordinatensystem verwendet, dessen x-Achse zum Mittelpunkt des Speicherringes, die y-Achse nach oben und die z-Achse in Flugrichtung der Protonen zeigt. Der Ursprung des Koordinatensystems ist mit dem nominellen Wechselpunkt identisch, also dem Punkt, in dem Elektronen und Protonen kollidieren sollen. Der Wechselwirkungspunkt ist in Abb. 12 durch ein Kreuz dargestellt.

Der Polarwinkel wird mit θ, der Azimutwinkel mit φ bezeichnet; θ wird gegen die Flugrichtung der Protonen gemessen.

3.2 Ein Beispielereignis

Die Detektor-Daten können z.B. mit dem Programm LAZE visualisiert werden. Die graphische Darstellung eines Ereignisses tiefunelastischer Streuung findet sich in Abb. 12. In der linken Hälfte ist eine Z-R Projektion des Detektors abgebildet. Bei dieser Art der Projektion kann die betrachtete Ebene nach Wunsch um das Strahlrohr gedreht werden. Der Detektor wird hierbei in zwei Hälften geteilt, in der Darstellung oben bzw. unten. Je Hälfte werden die Kalorimetereinträge bei gleicher z-Koordinate über den Azimutwinkel ϕ summiert und in eine einzige Zelle eingetragen. In dem zentralen Spurendetektor rekonstruierte Spuren werden bis zum Kalorimeter verlängert.

Rechts unten befindet sich eine Projektion der Spurenkammer auf die x,y-Ebene. Bei dieser Art von Projektion blickt man senkrecht zur Strahlrichtung in den Detektor, so daß für jedes darzustellende Objekt nur die x- und y-Koordinaten verwendet werden. Man erkennt die zwei Jets, die bei diesem Ereignis entstanden sind: ein Jet zeigt nach 'Nord-Osten' und einer fast direkt nach 'Süden' (siehe Abschnitt 6.1 für die Definition von Jets).

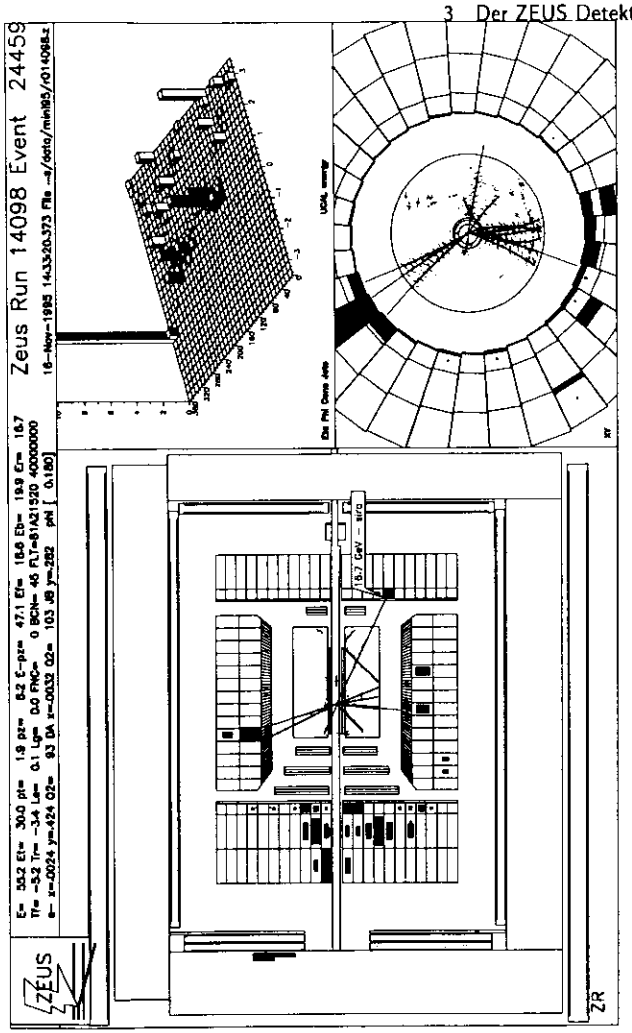


Abbildung 12: Mit dem ZEUS-Detektor registriertes Ereignis tiefunelastischer Streuung

Dem transversalen Gesamtimpuls der beiden Jets, der eine 'westliche' Komponente hat, steht das gestreute Positron in fast genau 'östlicher' Richtung gegenüber. Bis auf neutrale Teilchen muß der gesamte transversale Impuls sich zu Null summieren.

Rechts oben ist schließlich eine Projektion des Kalorimeters in die $\eta - \varphi$-Ebene zu finden. Hier erkennt man deutlich das Elektron (scharfes, hohes Maximum) und die zwei Jets (dunkel eingefärbt).

4 Auswertung der Detektorinformation

Durch den ZEUS-Detektor können Ereignisse mit einer Rate von ungefähr 20Hz aufgenommen werden, wobei große Mengen an Daten anfallen. Damit später effektiv auf die Daten zugegriffen werden kann, werden diese mittels eines relationalen Datenbanksystems, ADAMO ([16]), organisiert. Die Daten sind danach in Tabellen organisiert, die anhand verschiedener Relationen verknüpft sein können.

Auf ADAMO-Tabellen kann man mit Hilfe des Paketes **EAZE** (**E**ffortless **A**nalysis of **Z**eus **E**vents) zugreifen. Dieses Paket standardisiert den Zugriff auf die Daten, so daß dieser möglichst effizient gestaltet werden kann. Weiterhin vereinfacht sich das Schreiben eines Analyseprogrammes auf das Erstellen dreier Fortran-Routinen (**ZUINIT, ZUANAL, ZUTERM**), die in ein bestehendes Programmgerüst eingearbeitet (**gelinkt**) werden. In Abb. 13 ist die Struktur eines fertig gelinkten Analyseprogrammes aufgezeigt.

Links ist schematisch dargestellt, wie die eigenen Routinen in den bestehenden Programmrumpf eingefügt werden, rechts sieht man den Ablauf einer Analyse: Zu Beginn wird die Routine ZUINIT aufgerufen, hier koennen z.B. Dateien geöffnet und Variablen initialisiert werden. Anschließend wird für jedes Ereignis die Routine ZUANAL aufgerufen, in der der Benutzer einerseits entscheiden kann, ob er dieses Ereignis zur späteren Analyse festhalten möchte, es wird dann der komplette Datensatz des Ereignisses auf Festplatte geschrieben. Andererseits kann in dieser Routine auch eine Bearbeitung des Ereignisses stattfinden, falls man nur an bestimmten Elementen eines Ereignisses interessiert ist. So kann statt des kompletten Ereignisses zum Beispiel eine Datenstruktur auf Festplatte geschrieben werden, die abgeleitete Größen wie x, Q^2 eines Ereignisses und die Einträge des Kalorimeters enthält.

Die Routine ZUTERM wird genau einmal am Ende der Analyse aufgerufen und erlaubt dem Benutzer offene Dateien zu schließen und weiter

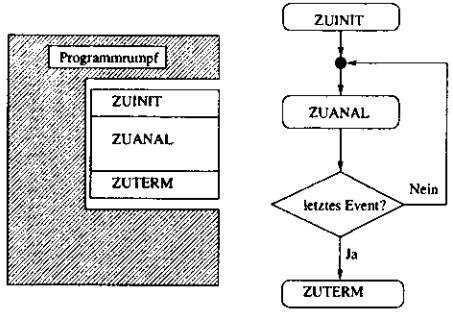


Abbildung 13: Ablauf eines Analyseprozesses

aufzuräumen.

Abb. 14 zeigt das Ergebnis einer solchen Auswertung, wobei die Daten durch das Programm PAW dargestellt wurden.

Die Abbildung zeigt das Energiespektrum von Elektronen, die aus der tiefunelastischen Streuung (DIS) kommen. Sie stammen aus Daten, die der ZEUS-Detektor 1995 aufgezeichnet hat und wurden durch die Forderung, daß das DST-Bit 14 gesetzt war (siehe Abschnitt. 7), aus ungefähr 1000 Ereignissen ausgewählt.

Die Elektronen wurden durch Aufruf des Programms Sinistra identifiziert.

Man erkennt sehr schön das Maximum in der Energie-Verteilung bei etwas unter 27GeV, das von Reaktionen mit geringem Energieübertrag auf das Proton (im Laborsystem) stammt, die Strahlenergie liegt bei ca. 27.5GeV.

Trägt man die Energie gegen die Winkel θ und ϕ auf, so entspricht die Verteilung der Erwartung: Die meisten Elektronen werden nur wenig von ihrer Bahn abgelenkt und unter einem Winkel von $\theta \approx \pi$ gestreut (θ wird gegen die „Vorwärts"-Richtung, also die Flugrichtung der Protonen, gemessen).

Bei ZEUS sind sowohl der Elektron- als auch der Protontrahl unpolarisiert, daher erwartet man eine flache Verteilung der Energie der gestreuten Elektronen in ϕ. In Abb. 14, rechts unten, wird diese Annahme bestätigt.

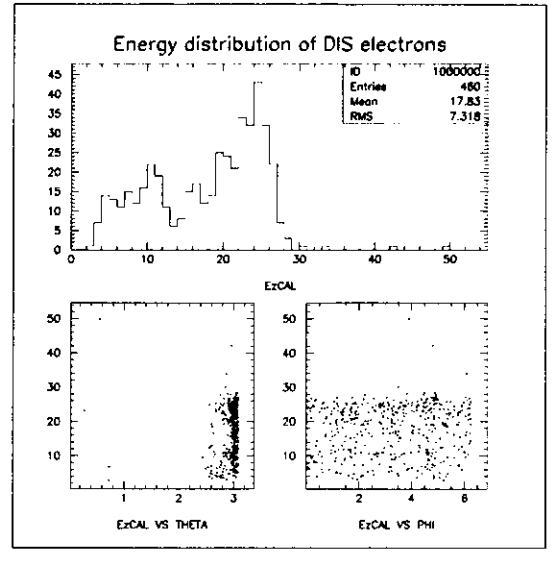


Abbildung 14: Energiespektrum von Elektronen aus tiefunelastischer Streuung, die durch Sinistra95 identifiziert wurden. *Oben* ist die Häufigkeit von rekonstruierten Elektronen gegen ihre Energie aufgetragen, *unten links* ist die Energie der Elektronen gegen ihren Polarwinkel θ aufgetragen, *unten rechts* ist die Energie der Elektronen gegen ihren Azimutwinkel aufgetragen. Man erkennt ein Maximum in der Häufigkeit der Energieeinträge bei knapp unter $27GeV$, also bei geringem Energieübertrag auf das Proton. Die gestreuten Elektronen werden am häufigsten bei großen Polarwinkeln nachgewiesen, wobei sie gleichmäßig über den Azimutwinkel verteilt sind.

5 Bestimmung von kinematischen Variablen

Die tiefunelastische Elektron-Proton Streuung (DIS=Deep Inelastic Scattering) muß im Gegensatz zu elastischer Streuung durch zwei kinematische Variablen beschrieben werden. Konventionell werden die Größen Q^2 und x dazu verwendet, bzw x und y. Diese Größen berechnen sich wie folgt [17]:

$$Q^2 = -(k-k')^2 = -q^2 \tag{17}$$

$$x = \frac{Q^2}{2Pq} \tag{18}$$

$$y = \frac{2Pq}{s} \tag{19}$$

$$\Rightarrow Q^2 = sxy \tag{20}$$

dabei ist: k, k' Viererimpuls des Elektrons vor bzw. nach dem Stoß, P Viererimpuls des einlaufenden Protons, $s = (P+k)^2$ Quadrat der Schwerpunktsenergie.

Die meßbaren Daten sind redundant in Bezug auf die Bestimmung je zwei unabhängiger kinematischer Größen ($\{Q^2,x\}$ bzw $\{x,y\}$), so daß unterschiedliche Methoden angewandt werden können, um diese zu rekonstruieren. Nach [18] gibt es im Wesentlichen drei Methoden:

- Die Elektron-Methode: Hier dienen Energie und Streuwinkel des Elektrons als Grundlage der Berechnungen.
- Die Jacquet-Blondel (JB) Methode: Das hadronische System liefert diesmal die Ausgangsbasis.
- Die Doppel-Winkel (DA=Double Angle) Methode berechnet die kinematischen Größen aus den Polarwinkeln des gestreuten Elektrons und des hadronischen Systems.

Abhängig von der jeweiligen Detektorauflösung bei verschiedenem x und Q^2 sind die Methoden unterschiedlich genau.

Bestimmt man die Größen komplett aus den Daten des Elektrons, so ergibt sich:

$$y = 1 - \frac{E'}{2E}(1-\cos\theta) \tag{21}$$

$$Q^2 = 2EE'(1+\cos\theta) \tag{22}$$

$$x = \frac{E}{E_p} \cdot \frac{E'(1+\cos\theta)}{2E - E'(1-\cos\theta)} \tag{23}$$

Dabei gilt:

E	...	Energie des Elektrons vor dem Stoß
E'	...	Energie des Elektrons nach dem Stoß
E_p	...	Energie des Protons vor dem Stoß
θ	...	Winkel des gestreuten Elektrons

Für die Bestimmung nach der DA-Methode wird außer dem Winkel des Elektrons noch der Winkel des hadronischen Systems γ benötigt. Im sogenannten naiven Quark-Parton-Modell (QPM) ist dies der Winkel, unter dem das Quark gestreut wird. Experimentell wird er wie folgt bestimmt:

$$\cos\gamma = \frac{p_T^2 - \delta_H^2}{p_T^2 + \delta_H^2}, \quad \text{mit} \tag{24}$$

$$p_T^2 = (\sum_i p_{iX})^2 + (\sum_i p_{iY})^2 \quad \text{und} \tag{25}$$

$$\delta_H = \sum_i (E_i - p_{iZ}) \tag{26}$$

Dabei läuft der Index i der Summen jeweils über alle Kalorimeterzellen, die nicht dem Elektron zugeordnet wurden. E_i ist die Energie der Kalorimeterzelle i, p_{iX}, p_{iY}, p_{iZ} sind die Projektionen des „Impulses" der Kalorimeterzelle auf die x- y- bzw z-Achse. Der „Impuls" einer Kalorimeterzelle ist eine fiktive Größe, die sich aus der Annahme ergibt, daß die in dieser Zelle gemessene Energiedeposition von genau einem Teilchen stammt. Aus dieser Energie und der Masse des Teilchens kann dann ein Impuls rekonstruiert werden. Diese Näherung wird häufig benutzt, in der Regel wird dabei entweder ein masseloses Teilchen oder z.B. ein π-Meson angenommen. In dieser Arbeit werden stets masselose Teilchen angenommen.

Die kinematischen Variablen y, Q^2, x ergeben sich nach der Doppelwinkel-Methode zu

$$y_{DA} = \frac{\sin\gamma(1-\cos\theta)}{\sin\gamma + \sin\theta - \sin(\theta+\gamma)} \tag{27}$$

$$Q^2_{DA} = 4E^2 \frac{\sin\gamma(1+\cos\theta)}{\sin\gamma+\sin\theta-\sin(\theta+\gamma)} \quad (28)$$

$$x_{DA} = \frac{E}{E_p}\cdot\frac{\sin\gamma+\sin\theta+\sin(\theta+\gamma)}{\sin\gamma+\sin\theta-\sin(\theta+\gamma)} \quad (29)$$

Für die Bezeichnungen s.o.
Nach der JB-Methode ergibt sich:

$$y_{JB} = \frac{\sum_i(E_i-p_{iz})}{2E} \quad (30)$$

$$Q^2_{JB} = \frac{(\sum_i p_{ix})^2+(\sum_i p_{iY})^2}{1-y_{JB}} \quad (31)$$

$$x_{JB} = \frac{Q^2_{JB}}{sy_{JB}} \quad (32)$$

Eine in Experimenten der Hochenergiephysik häufig verwendete Größe ist die Pseudorapidität η:

$$\eta = -\ln\tan\frac{\theta}{2} \quad (33)$$

dabei ist θ der Polarwinkel des gestreuten Teilchens gegen die Richtung des ursprünglichen Protons.

6 Jets

Die Idealvorstellung, bei Ereignissen der Hochenergiephysik direkt Impuls und Energie gestreuter Quarks und Gluonen zu messen, kann nicht erfüllt werden [19]. Der Grund hierfür ist, daß das Auftreten freier Quarks durch die Stärke der Quark-Quark-Kräfte verhindert zu sein scheint [20], die Stärke dieser Kräfte nimmt für hinreichend große Abstände annähernd linear mit dem Abstand zu. Dieses im Englischen „confinement" genannte Phänomen äußert sich in dem Entstehen neuer Quark-Antiquark Paare bei jedem Versuch, zwei Quarks mittels eines Stoßprozesses zu trennen. Es bildet sich eine hohe Energiedichte zwischen den zu trennenden Quarks aus, bis letztlich die Schwelle für Paarproduktion von Quarks überschritten wird. Da die Feldquanten der starken Wechselwirkung, die Gluonen, ebenfalls Farbladungen tragen, können auch sie nicht frei beobachtet werden.

Wird z.B. ein Quark eines Protons gestreut, so wird der Farbfluß zwischen diesem und dem Rest des Protons die Entstehung von neuen Quarkpaaren verursachen. Dieser Vorgang wird fortgesetzt, bis alle Quarks Partner gefunden haben, mit denen sie farbneutrale Objekte (wie z.B. π-Mesonen) bilden. Statt eines einzelnen gestreuten Quarks findet man in dessen Flugrichtung eine größere Anzahl von Hadronen, und auch im Bereich zwischen dieser Richtung und der Richtung des Protonrestes werden Hadronen nachgewiesen werden können. Dieser Prozeß wird **Fragmentierung** genannt.

Betrachtet man den Transversalimpuls der aus der Fragmentierung hervorgehenden Hadronen in Bezug auf die Flugrichtung des gestreuten Partons, so ist dieser gemessen an dem Impuls des Partons in der Regel gering. Infolgedessen findet man oft Teilchenbündel in der Flugrichtung eines gestreuten Partons, anhand dessen man auf die Kinematik des Partons schließen kann. Ein solches Teilchenbündel wird **Jet** genannt; für eine genauere Definition siehe z.B. [19].

6.1 Erkennung von Jets

Da die Rekonstruktion von Jets eine wichtige Voraussetzung für diese Arbeit ist, soll an dieser Stelle der verwendete Algorithmus beschrieben werden. Es handelt sich um den k_t-Algorithmus, der näher in [21] beschrieben ist.

Es gibt zwei grundlegende Arten von Jetfinder-Algorithmen, die in Ereignissen der tiefunelastischen Streuung eingesetzt werden:

- Cone-Algorithmen: Diese maximieren den Energiefluß durch Kreise mit konstantem Radius. Diese Definition von Jets wird Snowmass-Konvention genannt. Grundlage für die Rechnungen ist eine Projektion des Detektors auf die Ebene, die durch Azimutwinkel (φ) und Pseudorapidität (η) der Kalorimeterzellen aufgespannt wird. Der Radius ist als $R = \sqrt{\eta^2+\varphi^2}$ definiert, in der Regel wird $R = 0.7$ bzw. $R = 1.0$ gewählt.

- Cluster-Algorithmen: Diese definieren eine Abstandsnorm zwischen je zwei „Teilchen" und fügen in jeder Iteration die zwei Teilchen minimalen Abstandes zu einem neuen Teilchen zusammen, sofern dieser Abstand ein festgelegtes Maximum nicht überschreitet. Als „Teilchen" kommen rekonstruierte Teilchen in Frage, aber auch Kalorimeterzellen.

Die Wirkungsweise dieser Algorithmen ist in Abb. 15 graphisch dargestellt.

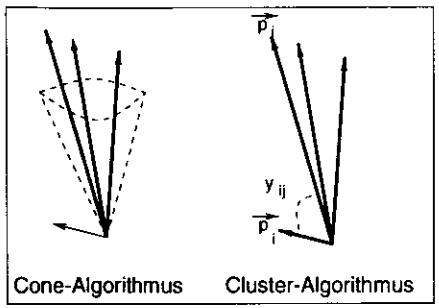


Abbildung 15: Funktionsweise der verschiedenen Jetfinder-Algorithmen. Man erkennt, daß Cone-Algorithmen systematisch Abstrahlungen geringer Energie unter großen Winkeln vernachlässigen. y_{ij} bezeichnet die Abstandsnorm der zwei Teilchen mit den Impulsen p_i und p_j. (s. Gl. 34)

Der k_t-Algorithmus ist ein Repräsentant der Cluster-Algorithmen, der verschiedene Abstandsnormen anbietet. Während dieser Arbeit wurde folgende Abstandsnorm verwendet:

$$y_{ij} = 2\frac{1 - cos\theta_{ij}}{Q^2_{DA}}\min(E_i^2, E_j^2) \tag{34}$$

mit

E_i, E_j	...	Energie der zwei Teilchen i, j
θ_{ij}	...	Winkel zwischen den Impulsvektoren der beiden Teilchen
Q^2_{DA}	...	Das mittels Doppel-Winkel-Methode rekonstruierte Q^2 des Ereignisses

Wendet man dieses Verfahren statt auf Teilchen auf Kalorimeterzellen an (indem man sie als masselose Teilchen betrachtet, siehe Abschnitt 5), wird anhand dieser Abstandsnorm für alle möglichen Paare von Kalorimeterzellen der Abstand berechnet. Die zwei Zellen, für die der Abstand minimal ist, werden zu einer Einheit verschmolzen. Diese Verschmelzung kann zum Beispiel durch einfache Addition der Viererimpulse stattfinden, dies ist das sog. *E-Schema*: $p_k = p_i + p_j$. i und j sind die zu verschmelzenden Teilchen, k ist das Verschmelzungsprodukt. Die Verschmelzung kann auch zum Beispiel durch Addition der Dreierimpulse $\vec{p}_k = \vec{p}_i + \vec{p}_j$ geschehen, während die Energie als $E_k = |\vec{p}_k|$ gesetzt wird [22]. In dieser Arbeit wird das E-Schema verwendet.

Ist für kein Paar der Abstand kleiner als ein vorher festgelegter Abschneideparameter y_{cut}, wird das Verfahren abgebrochen. Die verbliebenen „Teilchen" sind die rekonstruierten Jets.

In dieser Form wurde der Algorithmus ursprünglich in der e^+e^--Streuung verwendet, in der ep-Streuung muß zusätzlich der Protonenrest berücksichtigt werden. Im Falle der Cone-Algorithmen wird dafür bei einem Wert von η geschnitten, der k_t-Algorithmus kann mit *allen* Kalorimeterzellen arbeiten. Dazu wird zusätzlich eine Abstandsnorm zur Richtung des Protons definiert:

$$d_i = 2E_i^2(1 - \cos\theta_{ib})$$

Dabei ist θ_{ib} der Polarwinkel des Teilchens in Bezug auf die ursprüngliche Flugrichtung des Protons. Ist der Wert d_i geringer als jeder einzelne Teilchen-Teilchen-Abstand y_{ij}, dann wird das Teilchen zum Protonenrest gezählt.

7 Datenselektion

Zum Trainieren und Testen des neuronalen Netzes wurden durch das Programm Django 6.22 simulierte Ereignisse verwendet, eine ausführliche Beschreibung dieses Programms findet sich in [23].

Es handelt sich bei diesen Ereignissen um simulierte tiefunelastische Positron-Proton-Kollisionen, bei denen sowohl QED- als auch QCD-Strahlungseffekte berücksichtigt sind.

Bei dieser Analyse wurden nur Ereignisse betrachtet, für die das DST-Bit 14 gesetzt war, welches tiefunelastische Prozesse auswählt. Es umfaßt folgende Schnitte [24]:

- $E - p_z + 2E_{lumi} > 25$ GeV, wenn kein Vertex rekonstruiert werden konnte.
- $E - p_z + 2E_{lumi} > 35$ GeV, wenn ein Vertex rekonstruiert werden konnte.
- $E_{electron} > 5$ GeV, mindestens ein Elektron-Finder (ELEC oder LOCAL) muß fündig geworden sein
- $y_{JB} > 0.02$

- Das rekonstruierte Elektron muß in x oder y weiter als 16cm von der Strahlrohrmitte entfernt rekonstruiert worden sein.
- Die Programme RMSPARK, MUTRIG, ALHALO müssen überprüft haben, daß
  - nicht ein Funkendurchschlag in einer der Zellen des Kalorimeters den Trigger ausgelöst hat (RMSPARK)
  - der Trigger wahrscheinlich nicht durch kosmische und Halo-Muonen ausgelöst wurde. (MUTRIG und ALHALO)
- Der energiegewichtete Mittelwert der Ankunftszeit im RCAL muß innerhalb eines Fensters von 8ns vor bis 8ns nach dem berechneten Zeitpunkt des Ereignisses registriert worden sein.

Ferner wurden folgende Schnitte vorgenommen:

- $Q^2_{DA} \geq 20GeV^2$
- $x_{DA} \geq 0.001$
- $E_{elec} \geq 5GeV$
- $y_{JB} \geq 0.05$

Aus diesen Daten wurden zwei unabhängige Datensätze zum Training des neuronales Netzes generiert, ein Datensatz zum Trainieren des Netzes und einer, um den Lernerfolg des Netzes zu kontrollieren. Der Trainingsdatensatz umfaßte 2057 Ereignisse, der Testdatensatz 2887.

7.1 Energieverteilung von Jets

Zur Bestimmung eines für diese Analyse geeigneten Koordinatensystems soll zunächst der Energiefluß von Jets untersucht werden.

Es werden dabei in Monte-Carlo-Rechnungen erstellte Ereignisse auf ihre Jet-Struktur untersucht, siehe Abschnitt 7 für eine Beschreibung des Simulationsprogramms. Für jedes Ereignis wird zuerst ein Elektron, danach in den verbleibenden Kalorimeterzellen nach Jets gesucht. Zum Einsatz kommt hier zum Finden von Elektronen das Programm `Sinistra`, zur Bestimmung der Jets der k_t-Algorithmus, der in Abschnitt 6.1 vorgestellt wurde.

Gemessen wurde die zur z-Achse transversale Energieverteilung (E_T) der Jets als Funktion von $\Delta\eta$ und $\Delta\phi$. Abb. 16 zeigt dies, die Histogramme der transversalen Energie sind normiert auf die Anzahl der Ereignisse und aufgetragen gegen $\Delta\eta$ bzw. $\Delta\varphi$. $\Delta\eta$ und $\Delta\varphi$ ergeben sich zu

$$\begin{aligned} \Delta\eta &= \eta - \eta_{jet} \\ \Delta\varphi &= \varphi - \varphi_{jet} \end{aligned}$$

diese Größen und E_T werden für jede Kalorimeterzelle oberhalb der Energieschwellen aus Abschnitt 3 berechnet; η und φ sind Pseudorapidität und Azimutwinkel einer jeden betrachteten Kalorimeterzelle, η_{jet} und φ_{jet} sind Pseudorapidität und Azimutwinkel des durch den k_t-Algorithmus rekonstruierten Jets.

Betrachtet wurden 1- und 2-Jet-Ereignisse. Dabei bedeutet die Notation (1+1) einen Jet zzgl. zum Protonenrest, (2+1) zwei Jets zzgl. zum Protonenrest. Im folgenden soll der Protonenrest nicht mitgezählt werden, ein 1-Jet Ereignis ist demnach ein (1+1)-Ereignis. Für die 2-Jet Ereignisse werden $\Delta\eta$ und $\Delta\varphi$ einmal für den bei höherem η rekonstruierten Jet und einmal für den bei niedrigerem η rekonstruierten berechnet.

Die Schnitte, die hier verwendet wurden, orientieren sich an [25]. Folgende Einschränkung wurde zusätzlich zu den in Abschnitt 7 genannten gemacht:

- $20GeV^2 < Q^2_{DA} \leq 1280GeV^2$

Links oben in Abb. 16 ist der Energiefluß von 1-Jet-Ereignissen gegen $\Delta\varphi$ aufgetragen, Schraffiert ist der Energiefluß der zum Jet zugehörigen Zellen aufgetragen, in Weiß der Fluß der verbleibenden Zellen. Rechts oben ist der Energiefluß in $\Delta\eta$ zu sehen. Man erkennt einen Ausläufer der Verteilung zu hohem $\Delta\eta$, dies resultiert aus dem Energiefluß zwischen gestreutem Parton und Protonenrest.

In den unteren vier Darstellungen ist die entsprechende Darstellung für 2-Jet-Ereignisse zu finden. Die mittlere Reihe zeigt den Energiefluß für den Jet mit dem niedrigeren η-Wert, die untere Reihe bezieht sich auf den Jet bei größeren η-Werten.

Man erkennt in dieser Darstellung, daß Jets sowohl in $\Delta\varphi$ als auch in $\Delta\eta$ eine Ausdehnung von ungefähr 1 besitzen. Um diese Tatsache auszunutzen, wird in dieser Arbeit stets eine Projektion des Detektors in die $\Delta\eta - \Delta\varphi$-Ebene betrachtet.

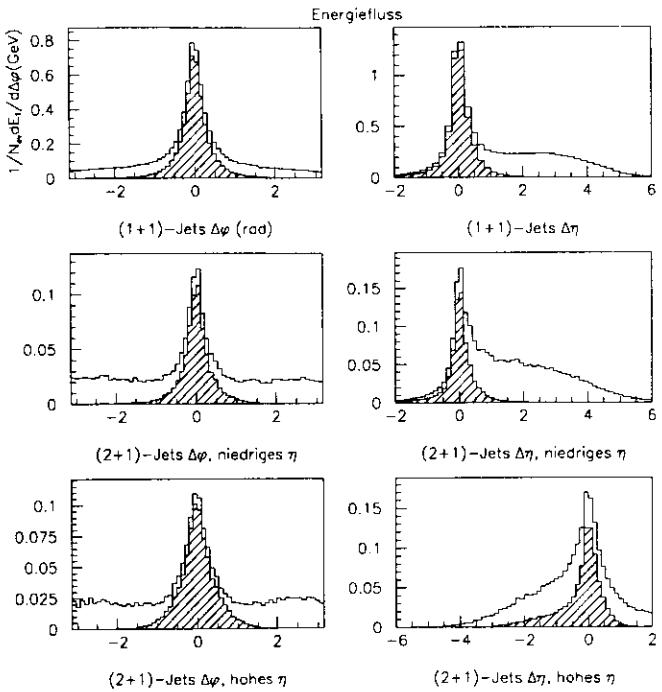


Abbildung 16: Fluß der transversalen Energie als Funktion von $\Delta\varphi$ und $\Delta\eta$. Schraffiert ist der Energiefluß der zum Jet zugehörigen Zellen auftragen. *o.l.:* Energiefluß von 1-Jet-Ereignissen, aufgetragen gegen $\Delta\varphi = \varphi - \varphi_{jet}$, φ_{jet} ist der Azimutwinkel des rekonstruierten Jets, *o.r.:* Energiefluß von 1-Jet-Ereignissen, aufgetragen gegen $\Delta\eta = \eta - \eta_{jet}$, η_{jet} ist die Pseudorapidität des rekonstruierten Jets, *m.l.:* Energiefluß gegen $\Delta\varphi$ in Bezug auf den zu niedrigerem η rekonstruierten Jet, *m.r.:* Energiefluß gegen $\Delta\eta$ in Bezug auf den zu niedrigerem η rekonstruierten Jet, *u.l.:* Energiefluß gegen $\Delta\varphi$ in Bezug auf den zu höherem η rekonstruierten Jet, *u.r.:* Energiefluß gegen $\Delta\eta$ in Bezug auf den zu höherem η rekonstruierten Jet

8 Auswahl der Variablen

Die Eingabevariablen für das neuronale Netz wurden aus vier verschiedenen Gebieten gewählt:

1. Winkel des gestreuten Partons im QPM
2. Globale 'Event Shape'-Variablen
3. Verteilung der transversalen Energiedeposition E_T
4. Anzahl der angesprochenen Zellen des Kalorimeters.

8.1 Winkel des gestreuten Partons

In Anlehnung an das Quark Parton Modell wird als Bezugsrichtung für die Ausbildung von Jets, insbesondere bei 1-Jet-Ereignissen, die Richtung (φ_0, η_γ) angenommen. Der Winkel φ_0 ist dabei der dem gestreuten Elektron gegenüberliegende Azimut-Winkel, η_γ ist die Pseudorapidität des Polarwinkels des gestreuten Partons und ergibt sich aus γ gemäß $\eta_\gamma = -\ln\tan\frac{\gamma}{2}$. Der Winkel γ selbst wird gemäß Gleichung 24 berechnet.

Auch bei 1-Jet-Ereignissen wird dieser Winkel nicht exakt in Richtung des Jets liegen, zumal im QPM der Farbfluß zwischen dem gestreuten Parton und den Spektatorquarks vernachlässigt wird.

Aus zwei Gründen werden im folgendenden alle Kalorimeterzellen gegen $\Delta\eta = \eta - \eta_\gamma$ und $\Delta\varphi = \varphi - \varphi_0$ aufgetragen: Einerseits variiert so die Lage des Ursprungs der betrachteten $\Delta\eta - \Delta\varphi$-Ebene von Ereignis zu Ereignis, so daß die quantisierte Struktur des Detektors verwischt wird. Andererseits ist dies eine gute Richtung, um mit der Suche nach Jets zu beginnen, da sie häufig nahe an dieser Richtung liegen.

8.2 Ereignis-Struktur-Variablen

Die Ereignis-Struktur ('Event Shape') Variablen wurden ursprünglich in der Elektron-Positron-Annihilation benutzt, wo sie ein maßgebliches Werkzeug bei der Suche nach Ereignissen mit drei Jets waren, die zur Entdeckung des Gluons führten [26].

Der Vorteil dieser Variablen liegt in Zusammenhang mit neuronalen Netzen darin, daß globale Aspekte des gesamten Ereignisses durch wenige Variablen beschrieben werden können.

Allerdings konnten diese Variablen nicht in ihrer ursprünglichen Bedeutung verwendet werden, da sich die Ereignisse der Positron-Proton-Streuung von denen der Elektron-Positron-Annihilation wesentlich unterscheiden. Insbesondere tritt bei tiefunelastischer Streuung der Protonenrest auf, der fragmentiert und als Untergrundsignal im Detektor registriert wird. Außerdem wurde in der Elektron-Positron-Annihilation stets nur eine Hemisphäre eines jeden Ereignisses zur Berechnung der Variablen verwendet. In dem hier gewählten Ansatz wurde von einer Unterteilung des Ereignisses in verschiedene Hemisphären abgesehen.

Die 'Event Shape' Variablen sind:

- Sphericity S
- Aplanarity A
- Thrust T

Die Größen S und A werden aus dem Impulstensor aller Teilchen im Endzustand gewonnen. Da bei der Entstehung von Jets einzelne Hadronen nicht auflösbar sind, wurden hier statt dessen die Kalorimeterzellen verwendet. Das Verfahren beruht auf einer Diagonalisierung des Impulstensors, analog der Berechnung der Trägheitsmomente. Aus den Eigenwerten werden die zwei unabhängigen Linearkombinationen S und A gebildet. Da die Summe der Eigenwerte auf eins normiert ist, gibt es keine weitere unabhängige Linearkombination.

Der Impulstensor ist gemäß [26] definiert als:

$$M_{\alpha\beta} = \sum_{j=1}^{N} p_{j\alpha} p_{j\beta} \qquad (\alpha, \beta = x, y, z)$$

Dabei läuft die Summe über alle N Teilchen (Kalorimeterzellen). Seien nun $\hat{n}_1, \hat{n}_2, \hat{n}_3$ die Einheitseigenvektoren dieses Tensors, assoziiert mit dem kleinsten (Λ_1), mittleren (Λ_2) und größten (Λ_3) Eigenvektor. Die Haupt-Jet-Achse ist dann in der $\hat{n}_3$-Richtung (**Sphärizitätsachse**), die Ereignisebene ist die $\hat{n}_2 - \hat{n}_3$-Ebene und $\hat{n}_1$ definiert die Richtung, in der die Summe der quadratischen Impulskomponenten minimal ist.

Aus den Eigenwerten lassen sich die normierten Größen

$$Q_k = \frac{\Lambda_k}{\sum_{j=1}^{N} \vec{p}_j^{\,2}}$$

bilden, aus denen sich wiederum die Aplanarität A und die Spherizität S ergeben zu:

$$A = \frac{3}{2} Q_1$$
$$S = \frac{3}{2}(Q_1 + Q_2)$$

Es gibt keine dritte unabhängige Linearkombination aus den Q_k, da diese normiert sind.

Anhand der Einheitsvektoren lassen sich noch weitere Variablen definieren. In Bezug auf die Ereignisebene $\hat{n}_2 - \hat{n}_3$ betrachtet man Impulse **in** bzw. **außerhalb** dieser Ebene.

Hieraus werden die folgenden Größen gebildet:

$\overline{p_t^2}$	Mittel der Summe der Quadrate der zur Spherizitätsachse orthogonalen Impulskomponenten aller Teilchen.
$\overline{p_{t,out}^2}$	Mittleres p_t^2 außerhalb der Ereignisebene
$\overline{p_{t,in}^2}$	Mittleres p_t^2 in der Ereignisebene
$\cos(\theta_s)$	Kosinus des Polarwinkels der Spherizitätsachse

Diese Größen wurden in [26] und [27] verwendet, um die Entstehung von Gluonen in der e^+e^--Annihilation nachzuweisen.

Der Thrust T wird anhand eines Maximierungsverfahrens gebildet und stellt damit wahrscheinlich den zeitintensivsten Aspekt der Berechnung der Eingabevariablen dar. Der Thrust ist definiert als [20]:

$$T = \max \frac{\sum_i |p_{li}|}{\sum_i |\vec{p}_i|}$$

Die Summe läuft über alle Teilchen, dabei sind die $\vec{p}_i$ die Impulse der Teilchen und die p_{li} ihre Impulskomponenten längs einer Achse, die so gewählt ist, des T maximal wird.

Es ist noch anzumerken, daß diese Größen in der tiefunelastischen ep-Streuung nicht entsprechend ihrem ursprünglichen Sinn verwendet werden können, zumal der Protonenrest nicht sinnvoll in die Definitionen aufgenommen werden kann. Das Trennungsvermögen des hier trainierten Netzes legt jedoch nahe, daß diese Variablen Diskriminierungspotential bezüglich der untersuchten Ereignisklassen bieten.

8.3 Verteilung der transversalen Energie E_T

Ein weiterer Ansatzpunkt ist die Verteilung der transversalen Energie E_T. Diese wird anstelle der Energie E verwendet, um den Einfluß des Protonenrests zu unterdrücken. Bestandteile des Rests werden in der Regel mit hoher Gesamt- aber geringer transversaler Energie im FCAL nachgewiesen.

Die Verteilung der Energie wurde im $\Delta\eta, \Delta\varphi$-Raum analysiert, da die Größe eines Jets in diesem Raum in beiden Dimensionen ungefähr eins ist. Als Ursprung wurde die Richtung (η_γ, φ_0) gewählt, wobei η_γ die Pseudorapidität des gestreuten Partons im QPM ist, während φ_0 der dem gestreuten Elektron gegenüberliegende Azimutwinkel ist.

Aus der Verteilung der transversalen Energie E_t wurden fünf Variablen gewonnen:

1. $\frac{\sum E_t \text{ in } 0<|\Delta\eta|<1}{\sum E_t \text{ in } 2<|\Delta\eta|<3}$
   Dies ist der Quotient der summierten Energiedeposition in den Pseudorapiditätsbereichen $|\Delta\eta| = 0..1$ und $|\Delta\eta| = 2..3$.
2. $\sum E_t$ in $0 < R < 1$, $R = \sqrt{\eta^2 + \varphi^2}$
3. $\frac{\sum E_t \text{ in } 0<R<1}{\sum E_t \text{ in } 2<R<3}$
4. $\overline{\varphi^2}$, berechnet gemäß
   $$\overline{\varphi^2} = \frac{\sum E_t(\varphi - \bar{\varphi})^2}{\sum E_t}$$
   mit $\bar{\varphi} = \frac{\sum E_t \varphi}{\sum E_t}$. Die Summen laufen stets über alle Kalorimeterzellen.
5. $\overline{\varphi^2}$, wobei nur diejenigen Kalorimeterzellen in die Summe eingehen, deren E_t mindestens doppelt so groß ist wie das Mittel aller Kalorimterzellen $\bar{E}_t$, die eine Deposition oberhalb der Schwellenwerte registrierten (siehe Kapitel 8.4).

8.4 Anzahl der Kalorimeterzellen

Als ein natürliches Maß für die Verteilung eines Ereignisses im Detektor wurde die Anzahl der Kalorimeterzellen als eine weitere Variable gewählt. Gezählt wurden alle Zellen mit einer Energie $> 110MeV$ in den HAC-Sektionen und $> 60MeV$ in den EMC-Sektionen.

9 Training des Netzes

9.1 Aufgabe

Dem neuronalen Netz wurde die Aufgabe gestellt, zwischen zwei Klassen von Ereignissen zu unterscheiden. Um die „Wahrheit", d.h. die Referenz zu definieren, anhand derer das Netz trainiert und getestet werden sollte, wurde der Jetfinder `ktclus`, der k_t-Algorithmus, eingesetzt. Die Klassen, die es zu unterscheiden galt, sind

A: Ereignisse mit 0 oder 1 Jets,

B: Ereignisse mit 2 oder mehr Jets.

Die Aussage des neuronalen Netzes wird ausschließlich mit dem k_t-Algorithmus verglichen, bei der Anwendung muß dies unbedingt berücksichtigt werden.

Bei den Ereignissen mit 0 Jets handelt es sich um solche, bei denen anhand der Werte von Q^2 und x ersichtlich ist, daß das gestreute Parton unter ungefähr 90° mit geringem Transversalimpuls auf das BCAL trifft. In einem solchen Fall kann es vorkommen, daß die zugehörigen Energiedepositionen auf Grund des geringen Transversalimpulses durch den Jetfinder fälschlicherweise dem Protonenrest zugeordnet werden.

9.2 Trainingsverlauf

Mit den 2057 Trainings- und 2887 Testmustern wurde ein neuronales Netz trainiert. Es handelt sich um ein Perceptron mit

- 14 Eingabeknoten:

1.	Sphericity S	8.	$\frac{\sum E_t \text{ in } 0<\lvert\eta\rvert<1}{\sum E_t \text{ in } 2<\lvert\eta\rvert<3}$
2.	Aplanarity A	9.	$\sum E_t$ in $0 < R < 1$
3.	Thrust T	10.	$\frac{\sum E_t \text{ in } 0<R<1}{\sum E_t \text{ in } 2<R<3}$
4.	$\overline{p_t^2}$	11.	$\overline{\varphi^2}$
5.	$\overline{p_{t,out}^2}$	12.	$\overline{\varphi^2}$, für $E_t > 2\bar{E}_t$
6.	$\overline{p_{t,in}^2}$	13.	Winkel des gestreuten Partons γ
7.	$\cos(\theta_s)$	14.	Anzahl der Kalorimeterzellen

- 5 versteckten Knoten und
- 1 Ausgabeknoten.

Der Trainingsvorgang wurde mit dem Programm dEXTRa durchgeführt, und zwar über 50000 Iterationen, anschließend wurde das Ergebnis des Trainings anhand der Testmuster überprüft. Der Trainingsverlauf ist in Abb. 17 abgebildet.

Aufgetragen sind wie in Abb. 7 die Temperatur (TEMPERAT) und die Fehler für die Trainingsmuster (TRAINERR) und für die Testmuster (TESTERR), als Abszisse dient die Nummer der Trainingsiteration (UPDATE).

Der Fehler berechnet sich zu:

$$\left.\begin{matrix}\text{TRAINERR}\\ \text{TESTERR}\end{matrix}\right\} = \frac{1}{2N}\sum_{\mu=1}^{N}[\Theta(O^\mu) - \zeta^\mu]^2 \tag{35}$$

Dabei ist N die Anzahl der Trainings- bzw. Testmuster, μ läuft über alle Muster $\mu = 1..N$, ζ^μ ist die Referenzantwort für Muster μ und die Funktion $\Theta(O^\mu)$ ist definiert als:

$$\Theta(O^\mu) = \begin{cases} +1, & O^\mu > 0.5 \\ 0, & O^\mu \leq 0.5 \end{cases} \tag{36}$$

Man sieht, daß die Temperatur nicht ganz auf Null fällt. Dies ist vermutlich auf die schwere Trennbarkeit der zwei Klassen zurückzuführen, da die Änderung der Temperatur selbst von dem Trainingserfolg des Netzwerkes abhängt.

Rechts unten ist die Verteilung der Ausgabewerte des Netzes abgebildet. Die gestrichelte Linie stellt die Ereignisse dar, bei den die Antwort des

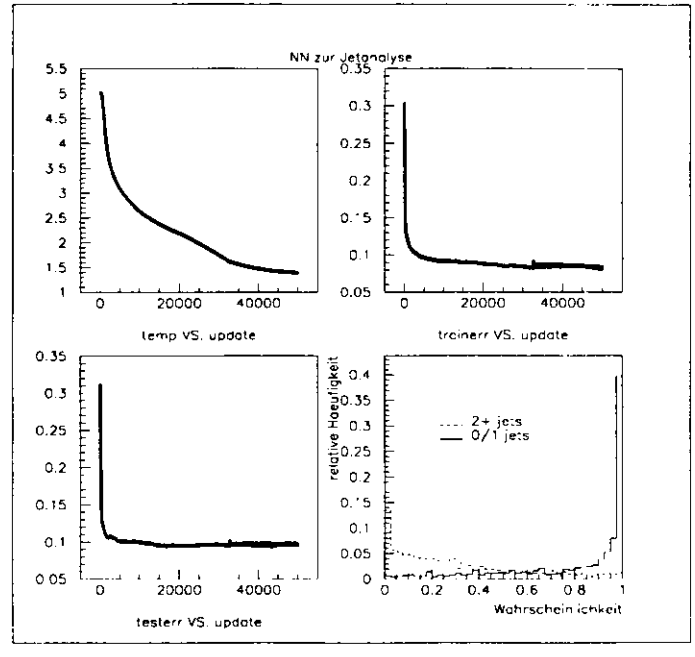


Abbildung 17: Trainingsverlauf mit den 14 ursprünglichen Variablen. Als Abszisse dient die Nummer der Trainingsiteration (update), aufgetragen sind die Temperatur (temp) und die Fehler für den Trainings- (trainerr) und den Testdatensatz (testerr). Für die Berechnung der Fehler siehe Gl. 35. Rechts unten ist die Verteilung der Ausgabewerte des Netzes dargestellt, die durchgezogene Line wäre im Idealfall ein scharfes Maximum um 1, die gestrichelte ein scharfes Maximum bei 0.

Netzwerkes '0' sein sollte, einem 2-oder-mehr-Jet Ereignis entsprechend. Die durchgezogene Linie hingegen entspricht den 0/1-Jet Ereignissen, bei denen eine Antwort '1' des Netzes richtig gewesen wäre.

10 Hauptachsenanalyse

Die Wahl der Eingabevariablen für ein Neuronales Netzwerk beruht stark auf Intuition, da kein standardisiertes Verfahren für diesen Schritt existiert. In der Regel wird man bei den ersten Versuchen, ein Neuronales Netzwerk für eine bestimmte Aufgabe zu trainieren, mehr Variablen als benötigt verwenden. Erfüllt das Netzwerk seine Aufgabe, gilt es herauszufinden, welche Variablen im Sinne des Entscheidungsprozesses überflüssig sind.

Ein Verfahren, mit dem die Relevanz einzelner Eingabevariablen überprüft werden kann, ist die sog. Hauptachsenanalyse (Principal Component Analysis [28], PCA). Das Prinzip beruht darauf, Linearkombinationen der Eingabevariablen zu finden, so daß die neuen Variablen keine Korrelation aufweisen.

Des weiteren können diese Variablen nun nach ihren Varianzen geordnet werden, unter der Annahme, daß die Variablen mit der größten Varianz auch am besten zur Diskriminierung geeignet sind. Dabei müssen alle Eingabevariablen stets in der gleichen Größenordnung sein, am besten sogar normiert.

Das Verfahren ist ausführlich in [28] beschrieben und wurde bereits in [8] und [29] verwendet.

Das Verfahren besteht im Wesentlichen aus zwei Transformationen:

- einer **Translation**, die eine Verschiebung zum Schwerpunkt der Variablen durchführt und
- einer **Rotation**, die eine Hauptachsentransformation durchführt.

Verwendet wird hierbei die sog. Dispersionsmatrix, deren Eigenwerte gerade die Varianzen der transformierten Variablen sind.

10.1 Mathematische Grundlage

Die folgende mathematische Behandlung der PCA ist im Wesentlichen eine Übersetzung von [28].

Die Eingabevariablen lassen sich als N m-dimensionale Vektoren x_α darstellen, mit $\alpha = 1..N$. Gesucht ist eine lineare Transformation der Koordinaten, d.h. eine Rotation W und eine Translation $\vec{c}$:

$$\vec{\xi} = W(\vec{x} - \vec{c})$$

Zusätzlich wird gefordert, daß die Komponenten ξ_j der neuen Linearkombinationen $\vec{\xi}$ entsprechend der Varianzen ihrer Verteilungen geordnet seien. Die Komponente mit $j = 1$ möge die größte Varianz haben, die Komponente $j = j_{max}$ die geringste.

Unter der Bedingung, daß W eine eigentliche Rotation sei, wollen wir

$$\bar{\xi}_j^2 = \frac{1}{N} \sum_{\alpha=1}^{N} \xi_{j\alpha}^2 \tag{37}$$

minimieren, für $j = m, ..., 2, 1$ sukzessive. Der Index α läuft über alle N Muster.

Bestimmung der c_i geschieht durch:

$$\frac{\partial}{\partial c_i} \sum_{\alpha=1}^{N} \xi_{j\alpha}^2 = \frac{\partial}{dc_i} \sum_{\alpha=1}^{N} \left[\sum_k w_{kj}(x_{k\alpha} - c_k) \right]^2 = 0$$

Nach dem Ableiten bleibt von der Summe über k nur der Term mit $k = i$ übrig, so daß sich diese Bedingung reduziert auf $\sum_{\alpha=1}^{N}(x_{i\alpha} - c_i) = 0$ bzw. $c_i = \bar{x}_i$, mit

$$\bar{x}_i = \frac{1}{N} \sum_{\alpha=1}^{N} x_{i\alpha}$$

Dies bedeutet, daß der neue Ursprung im Schwerpunkt liegt. Im folgenden sei $\vec{u} = \vec{x} - \vec{c}$, so daß für alle N Punkte gilt

$$\vec{\xi} = W\vec{u}$$

bzw.

$$\xi_j^2 = (\sum_i w_{ij} u_i)^2$$

Damit W eine eigentliche Rotation sein kann, müssen wir verlangen, daß

$$\sum_{i=1}^{m} w_{ij}^2 = 1$$

Diese Bedingung wird anhand eines Lagrangeschen Multiplikators λ_j beim Minimieren von Gleichung 37 eingeführt. Statt

$$\frac{\partial}{\partial w_{ij}}\bar{\xi}_j^2 = \frac{\partial}{\partial w_{ij}}\left(\frac{1}{N}\sum_{\alpha}^{N}\xi_{j\alpha}^2\right) = 0$$

ist nun

$$\frac{\partial}{\partial w_{ij}}\left(\frac{1}{N}\sum_{\alpha}^{N}\xi_{j\alpha}^2 + \frac{1}{N}\lambda_j\underbrace{\left[1-\sum_{k}^{m}w_{kj}^2\right]}_{=0}\right) = 0$$

zu berechnen. Der Faktor $1/N$ vor dem λ_j wurde eingeführt, um folgende Schritte zu vereinfachen. Entfernt man von den w_{kj} unabhängige Terme und konstante Faktoren aus der Summe, so bleibt:

$$\frac{\partial}{\partial w_{ij}}\left(\sum_{\alpha}^{N}\xi_{j\alpha}^2 - \lambda_j\sum_{k}^{m}w_{kj}^2\right) = 0$$

Das führt zu

$$\begin{aligned}
\frac{\partial}{\partial w_{ij}}\left[\sum_{\alpha}^{N}\left(\underbrace{\sum_{k}^{m}u_{k\alpha}w_{kj}}_{\xi_{j\alpha}}\right)^2\right] &= 2\lambda_j w_{ij}\\
\frac{\partial}{\partial w_{ij}}\left[\sum_{\alpha}^{N}\underbrace{\sum_{k}^{m}\sum_{l}^{m}(u_{k\alpha}u_{l\alpha}w_{kj}w_{lj})}_{(\sum_k(\cdots))^2}\right] &= 2\lambda_j w_{ij}\\
\sum_{\alpha}^{N}\sum_{k}^{m}u_{k\alpha}u_{i\alpha}w_{kj} &= \lambda_j w_{ij} \qquad (38)
\end{aligned}$$

Sieht man die Matrixelemente w_{ij} als Komponenten von Vektoren $\vec{w}_j$ an, so kann man Gleichung 38 in Vektordarstellung auch wie folgt schreiben:

$$A\vec{w}_j = \lambda_j\vec{w}_j \qquad (39)$$

Dabei ergeben sich die Elemente von A, der sogenannten Dispersionsmatrix, nach Gleichung 38 zu:

$$\begin{aligned}
a_{ki} &= \frac{1}{N}\sum_{\alpha}^{N}u_{k\alpha}u_{i\alpha}\\
&= \frac{1}{N}\sum_{\alpha}^{N}(x_{k\alpha}-\bar{x}_k)(x_{i\alpha}-\bar{x}_i)
\end{aligned}$$

Aus Gleichung 39 erkennt man, daß w_j ein Eigenvektor von A ist. Außerdem ergeben sich die $\bar{\xi}_j^2$, also die mittleren Quadrate der $\xi_{j\alpha}$, zu:

$$\begin{aligned}
\bar{\xi}_j^2 = \frac{1}{N}\sum_{\alpha}^{N}\xi_{\alpha j}^2 &= \frac{1}{N}\sum_{\alpha}^{N}\sum_{k}^{m}\sum_{i}^{m}w_{ij}u_{i\alpha}w_{kj}u_{k\alpha}\\
&= \frac{1}{N}\sum_{i}^{m}w_{ij}\underbrace{\left(\sum_{\alpha}^{N}\sum_{k}^{m}u_{k\alpha}u_{i\alpha}w_{kj}\right)}_{=\lambda_j w_{ij}\ \text{s.Gl.38}}\\
&= \sum_{i}^{m}w_{ij}\lambda_j w_{ij} = \lambda_j\sum_{i}^{m}w_{ij}^2 = \lambda_j
\end{aligned}$$

Es lassen sich folgende Schlüsse ziehen:

1. die Translation findet zum Schwerpunkt statt;
2. die Rotation wird beschrieben durch die Matrix der Eigenvektoren der Dispersionsmatrix;
3. die mittlere quadratische Summe der $j-$ten Komponente der neuen Koordinaten entspricht dem $j-$ten Eigenwert der Dispersionsmatrix.

Da Eigenvektoren orthogonal sind (hier sogar orthonormal), ist die inverse Matrix zu W ihre transponierte. Demzufolge gilt nicht nur

$$\xi_j = \sum_{i}^{m}w_{ij}(x_i-\bar{x}_i)$$

sondern auch

$$x_i = \bar{x}_i + \sum_{j}^{m}w_{ij}\xi_j$$

10.2 Ergebnis der PCA

Aufgetragen in Abb. 18 sind die für die beiden Ereignisklassen „0/1 Jets“ und „2+ Jets“ bestimmte Varianz σ^2 der transformierten Variablen, die Differenz Δ der Varianzen und die relative Differenz δ gemäß folgenden Gleichungen:

$$\Delta = |\sigma^2_{2+\text{ Jets}} - \sigma^2_{0/1\text{ Jets}}|$$

$$\delta = \frac{\sigma^2_{2+\text{ Jets}} - \sigma^2_{0/1\text{ Jets}}}{\max\left(\sigma^2_{2+\text{ Jets}}, \sigma^2_{0/1\text{ Jets}}\right)}$$

Die Variablen mit Varianzen in der Größenordnung 1 haben auch Differenzen in dieser Größenordnung, diese werden den Hauptbeitrag bei der Diskriminierung liefern. Allerdings deutet die Tatsache, daß die relativen Differenzen der gleichen Größenordnung sind, darauf hin, daß alle Variablen einen wichtigen Beitrag liefern [8].

Die Transformationsmatrix W ist in Abb. 19 zu finden. Ein Spaltenvektor $\vec{\xi}$ transformierter Variablen ergibt sich durch einfache Matrixmultiplikation eines Spaltenvektors $\vec{x}$ ursprünglicher Variablen mit dieser Matrix:

$$\vec{\xi} = W \cdot \vec{x}$$

Die Zeilen der Transformationsmatrix sind dabei die nach aufsteigenden Eigenwerten geordneten Eigenvektoren der Kovarianzmatrix, d.h. der größte Eigenwert, Nr. 1 in Abb. 18, entspricht der *letzten* Zeile der Matrix. Die vier transformierten Variablen mit der größten Varianz entsprechen also den unteren vier Zeilen.

Die Matrixelemente geben gewissermaßen darüber Auskunft, welche Ursprungsvariablen die größte Signifikanz besitzen. Betrachtet man die Eigenvektoren, bei denen $|\delta| > 0.5$ ist, so ergeben sich aus diesen die 'wichtigsten' Variablen als diejenigen, die einen möglichst großen Beitrag liefern. Die entsprechenden Matrixelemente (größer als 0.20) sind in der Matrix durch einen Kasten hervorgehoben. Diese Variablen entsprechen ('Zeile' bedeutet Zeilennummer *von unten*):

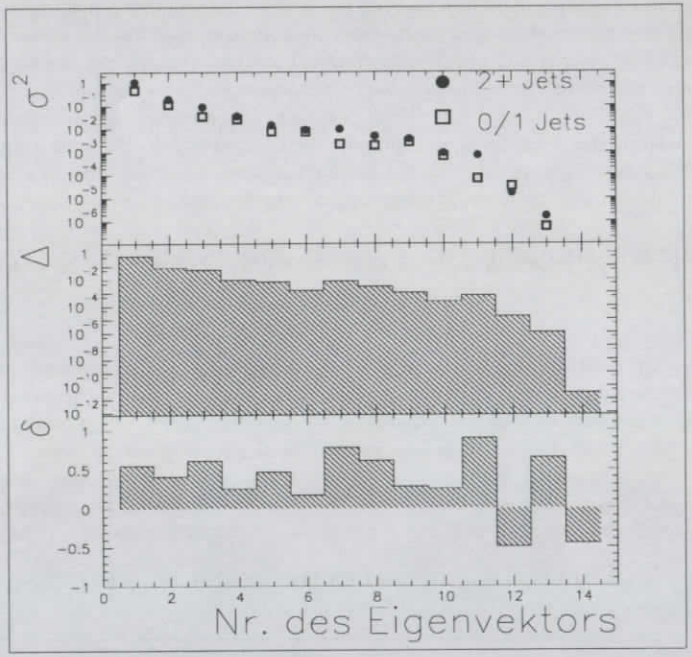


Abbildung 18: Ergebnis der Hauptachsentransformation: Eigenwerte der Dispersionsmatrix (σ^2 der Transformierten Variablen), Δ, δ. Wichtige Variablen zeichnen sich durch $|\delta| > 0.5$ aus.

$$
\begin{pmatrix}
0.000 & 0.000 & 0.001 & 0.009 & -.007 & 0.056 & 0.027 & 0.001 & 0.074 & 0.004 & -.781 & -.087 & \boxed{0.596} \\
-.001 & 0.001 & 0.001 & 0.031 & -.013 & 0.102 & 0.115 & \boxed{0.807} & -.031 & -.016 & 0.349 & 0.051 & \boxed{0.419} \\
-.006 & 0.009 & 0.002 & -.013 & -.080 & 0.032 & 0.019 & \boxed{-.524} & 0.030 & 0.338 & 0.478 & -.020 & \boxed{0.598} \\
0.000 & 0.000 & 0.000 & 0.000 & 0.013 & 0.032 & 0.020 & 0.051 & 0.021 & 0.001 & 0.022 & -.017 & 0.220 \\
0.000 & 0.013 & 0.023 & 0.011 & 0.975 & 0.111 & -.122 & 0.104 & 0.077 & 0.039 & -.001 & -.063 & 0.018 \\
0.000 & -.036 & \boxed{0.004} & \boxed{-.984} & 0.041 & -.053 & 0.012 & -.139 & -.069 & 0.018 & -.004 & 0.029 & -.008 \\
-.002 & \boxed{0.997} & 0.065 & -.035 & -.013 & 0.001 & 0.000 & -.011 & -.007 & -.002 & 0.000 & 0.002 & -.001 \\
0.000 & 0.006 & 0.001 & -.139 & -.079 & -.232 & 0.138 & \boxed{0.900} & 0.286 & 0.004 & -.017 & -.094 & -.002 \\
0.000 & 0.000 & 0.003 & -.002 & 0.004 & -.045 & -.017 & -.008 & 0.030 & -.042 & 0.537 & 0.087 & \boxed{0.084} \\
0.000 & 0.000 & -.005 & 0.001 & -.007 & 0.066 & 0.031 & 0.007 & -.051 & 0.072 & -.831 & 0.007 & \boxed{0.070} \\
-.577 & -.024 & 0.394 & 0.064 & 0.089 & -.667 & 0.057 & -.191 & 0.096 & -.002 & -.064 & 0.037 & -.002 \\
0.579 & -.052 & 0.813 & 0.006 & -.016 & -.016 & 0.000 & -.008 & 0.010 & -.009 & -.007 & -.002 & -.002 \\
0.577 & 0.032 & -.422 & 0.059 & 0.105 & -.652 & 0.057 & -.183 & 0.086 & 0.008 & -.057 & 0.039 & 0.000 \\
0.000 & 0.002 & -.013 & -.040 & -.044 & 0.198 & 0.174 & \boxed{.286} & 0.879 & -.172 & -.039 & \boxed{-.198} & -.040 \\
 & & & & & & & & & & & & -.021 \\
 & & & & & & & & & & & & 0.005 \\
 & & & & & & & & & & & & -.001 \\
 & & & & & & & & & & & & 0.004 \\
 & & & & & & & & & & & & \boxed{-.539} \\
 & & & & & & & & & & & & \boxed{-.832} \\
 & & & & & & & & & & & & -.010 \\
 & & & & & & & & & & & & 0.000 \\
 & & & & & & & & & & & & 0.003 \\
 & & & & & & & & & & & & -.004 \\
 & & & & & & & & & & & & -.089 \\
 & & & & & & & & & & & & 0.058 \\
 & & & & & & & & & & & & 0.052 \\
 & & & & & & & & & & & & 0.056
\end{pmatrix}
$$

Abbildung 19: Transformationsmatrix W der PCA. Die Koeffizienten der wichtigsten Linearkombinationen größer als 0.2 sind hervorgehoben.

Nr.	Variable	Zeile(n)
1	$\frac{\sum E_t \text{ in } 0<\lvert\eta\rvert<1}{\sum E_t \text{ in } 2<\lvert\eta\rvert<3}$	3
2	$\sum E_t$ in $0 < R < 1$	3,8
3	$\frac{\sum E_t \text{ in } 0<R<1}{\sum E_t \text{ in } 2<R<3}$	3,8
6	Sphericity S	11
7	Aplanarity A	13
8	$\cos(\theta_s)$	7
9	$\overline{\varphi^2}$	1
10	$\overline{\varphi^2}$, für E_t größer über $2\bar{E}_t$	1
14	Anzahl der Kalorimeterzellen	3,7

10.3 Training mit transformierten Variablen

Nach Durchführung der PCA wurden die vier Variablen mit der größten Varianz benutzt, um ein weiteres neuronales Netz zu trainieren. Das Ergebnis ist in Abb. 20 dargestellt.

Diese Abbildung ist mit dem Training mit allen 14 Variablen (Abb. 17 auf Seite 47) zu vergleichen: Das Trennvermögen hat zwar abgenommen, die Trennfähigkeit ist aber immer noch vergleichbar trotz stark reduzierter Zahl der Eingabeknoten. Ein ausführlicher Vergleich folgt in Abschnitt 12 Auf Seite 61.

11 Systematischer Ansatz zur Variablenwahl

Fragestellungen, die sich der direkten Anschauung entziehen, erschweren die Auswahl von geeigneten Eingabevariablen für ein neuronales Netz. In diesen Fällen muß man auf systematische Methoden zurückgreifen, um erfolgversprechende Eingabevariablen zu finden.

Im folgenden wird beschrieben, wie eine Fourier-Transformation des Kalorimeters des ZEUS-Detektors vorgenommen wurde und anhand eines systematischen Zugangs Variablen für ein neuronales Netz gefunden wurden. Die Aufgabe bestand erneut darin, ein Netzwerk zu erstellen, das zwischen Ereignissen mit keinem oder einem Jet und solchen mit zwei oder mehr Jets zu unterscheiden vermag.

Die Methoden zur Fouriertransformation des Detektors lehnen sich stark

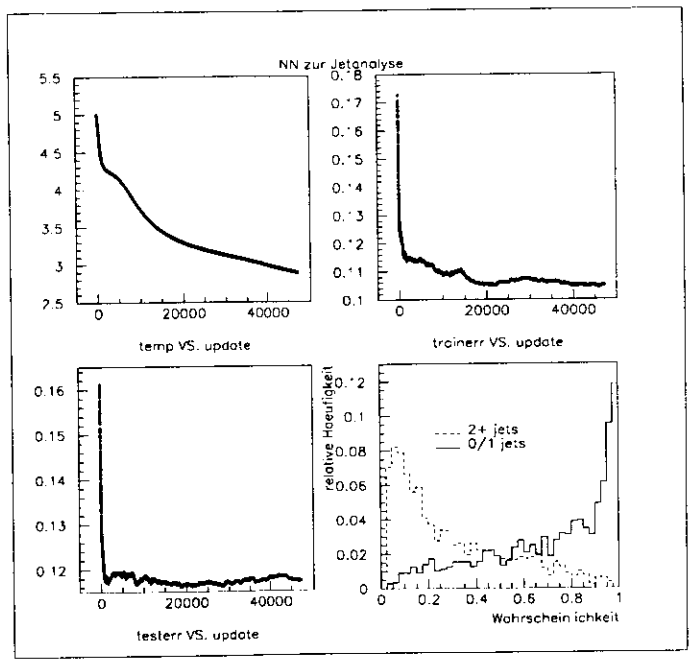


Abbildung 20: Trainingsverlauf mit vier transformierten Variablen. Als Abszisse dient die Nummer der Trainingsiteration (update), aufgetragen sind die Temperatur (temp) und die Fehler für den Trainings- (trainerr) und den Testdatensatz (testerr). Für die Berechnung der Fehler siehe Gl. 35. Rechts unten ist die Verteilung der Ausgabewerte des Netzes dargestellt, die durchgezogene Line wäre im Idealfall ein scharfes Maximum um 1, die gestrichelte ein scharfes Maximum bei 0.

an [30] an, es wird ein Algorithmus zur diskreten schnellen Fouriertransformation (discrete Fast Fourier transform) angewandt.

11.1 Fouriertransformation

Die Fouriertransformation nutzt die Orthonormalität der Funktionen *sin* und *cos*, um beliebige Funktionen nach diesen zu entwickeln, die dabei nach ihren Frequenzen zerlegt werden. In komplexer Schreibweise lautet die allgemeine Fouriertransformation:

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f t} dt$$

mit der Inversen

$$h(t) = \int_{-\infty}^{\infty} H(f) e^{-2\pi i f t} df$$

Dabei ist $h(t)$ die ursprüngliche Funktion und $H(f)$ die Fouriertransformierte. Man erhält auf diese Weise eine Entwicklung einer kontinuierlichen Funktion $h(t)$ in ein kontinuierliches Spektrum $H(f)$. In der Praxis ist die exakte Vermessung einer kontinuierlichen Funktion allerdings nicht möglich, man wird sie vielmehr an diskreten Punkten messen. Es erweist sich als günstig, N äquidistante Punkte h_k mit einem Abstand Δ auszuwählen ([30]), d.h.

$$h_k \equiv h(t_k), t_k = k\Delta, k = 0, 1, 2, \ldots, N-1$$

Dabei ist Δ das sogenannte **Abtastintervall**. Die Breite des Fourierspektrums ist durch dieses begrenzt, der Zusammenhang wird durch das **Abtasttheorem** beschrieben. Dieses besagt, daß das aufgenommene Spektrum auf einen Bereich $-f_{krit} \leq f \leq +f_{krit}$ beschränkt sein wird, mit der sog. **kritischen Frequenz** f_{krit}. Für f_{krit} gilt dann:

$$f_{krit} := \frac{1}{2\Delta}$$

Das Spektrum der Fouriertransformierten ist auch diskret und läßt sich anhand des Abtastintervalls Δ darstellen an den „Frequenzwerten"

$$f_n := \frac{n}{N\Delta}, n = -\frac{N}{2}, \ldots, \frac{N}{2}$$

Die diskrete Fouriertransformation lautet nun:

$$H_n := \sum_{k=0}^{N-1} h_k e^{2\pi ikn/N}$$

Die Rücktransformation geschieht mittels:

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\pi ikn/N}$$

Ein großer Vorteil der Fouriertransformation ist, daß das Potenzspektrum (Betrag der Fouriertransformierten) unabhängig von einer Translation im Ortsraum ist. Wendet man eine Translation $h(t) \mapsto h(t+\alpha)$ an, so schlägt sich diese nur in der Phase der Fouriertransformierten nieder, das Potenzspektrum bleibt davon unberührt.

Die während dieser Arbeit benötigte Fouriertransformation im zweidimensionalen Raum kann dargestellt werden als eine Verknüpfung zweier eindimensionaler Fouriertransformationen.

Hierfür wird der ZEUS-Detektor künstlich in 32x32 Segmente in der $\eta-\varphi$-Ebene unterteilt. Dadurch läßt sich die Detektorinformation in einer 32x32-Matrix unterbringen, die je $N = 32$ äquidistanten Abtastpunkten in η und in φ entspricht.

Der abgetastete Bereich entspricht:

$$\begin{aligned} \eta &\in [-4,4] \\ \varphi &\in [-\pi,\pi] \end{aligned}$$

Die Variablen im Frequenzraum, k_η und k_φ, decken gemäß $f_{krit} = \frac{1}{2\Delta}$, $\Delta_\eta = (\eta_{max} - \eta_{min})/32 = 1/4$ und $\Delta_\varphi = (\varphi_{max} - \varphi_{min})/32 = \pi/16$ folgende Bereiche ab:

$$\begin{aligned} k_\eta &\in [-2,2] \\ k_\varphi &\in [-\frac{8}{\pi},\frac{8}{\pi}] \end{aligned}$$

Es ergibt sich dann die transformierte H_{n_η,n_φ} aus der Ursprungsfunktion h_{i_η,i_φ} wie folgt:

$$H_{n_\eta,n_\varphi} = \sum_{j_\eta=0}^{N-1} \sum_{j_\varphi=0}^{N-1} h_{j_\eta,j_\varphi} e^{2\pi ij_\eta n_\eta/N} e^{2\pi ij_\varphi n_\varphi/N}$$

dabei sind n_η, n_φ Indizes, die jeweils von $0..N$ laufen.

11.2 Überprüfung der Transformationsroutine

Eine graphische Überprüfung der Transformationsroutinen findet sich in Abb. 21.

Die Abbildungen stellen ein Ein-Jet-Ereignis dar, wie es vom Kalorimeter des ZEUS-Detektors aufgenommen wurde. Oben links ist die Darstellung, in der das Darstellungsprogramm PAW die Einteilung in die verschiedenen Bins vornimmt, oben rechts wurde das Ereignis innerhalb des Programms in eine Matrix mit 32^2 Elementen geschrieben. Unten links ist der Betrag der Fouriertransformierten (das Potenzspektrum) dieser Matrix zu sehen. Die eigentliche Überprüfung besteht in der Rücktransformation, deren Ergebnis unten rechts zu sehen ist.

Sowohl die Zuordnung zu den Elementen der Matrix als auch die Fouriertransformation sind also korrekt implementiert worden.

Eine Beschreibung der schnellen Fouriertransformation (FFT) sowie der Fortran-Quelltext für die verwendeten Routinen findet sich in [30].

11.3 Systematische Suche nach Variablen

Um nach passenden Eingabevariablen im zweidimensionalen Fourierraum zu suchen, werden z.B. die Verteilungen der fouriertransformierten transversalen Energie E_t analysiert. Dabei wird angenommen, daß an den Punkten im Frequenzraum besonders gute Möglichkeiten zur Unterscheidung der betrachteten Ereignisklassen bestehen, an denen sich die Mittelwerte der Verteilungen des Quadrates des Betrages der transformierten E_t für die beiden Klassen signifikant unterscheiden.

Die Mittelwerte ergeben sich zu:

$$< X_{01|23} >= \frac{1}{N} \sum_{\mu=1}^{N} |H_{n_\eta,n_\varphi}|^2$$

dabei erstreckt sich die Summe für den Mittelwert $< X_{01} >$ über alle Ereignisse des Trainingsdatensatzes mit höchstens einem Jet, für den Mittelwert $< X_{23} >$ über die Ereignisse mit zwei oder drei Jets, H_{n_η,n_φ} ist die Fouriertransformierte von E_t. Dieser Mittelwert wird für jedes der 32x32 Index-Paare (n_η, n_φ) gebildet.

Aus den Mittelwerten wird die Testgröße δ gebildet, die eine gewichtete Differenz dieser Mittelwerte darstellt:

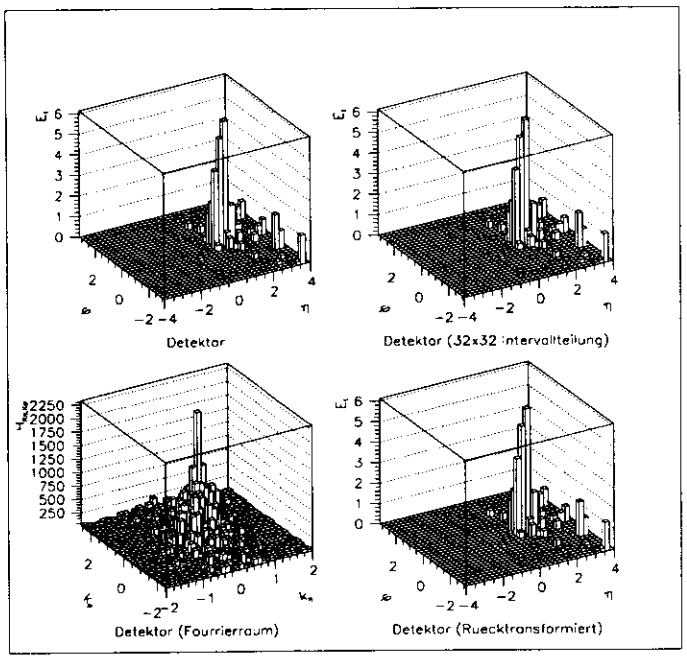


Abbildung 21: Darstellung eines 1-Jet Ereignisses in Orts- und Fourierraum. *lo:* Transversale Energie gegen φ (Azimut) und η (Peudorapidität); *ro:* 32x32 Intervalleinteilung im Ortsraum; *lu:* Fouriertransformierte, aufgetragen ist $|H_{k_\eta,k_\varphi}|$; *ru:* Rücktransformierte im Ortsraum, vgl. mit *ro.*

$$\delta = \frac{< X_{01} > - < X_{2+} >}{\sqrt{\sigma^2_{X_{01}} + \sigma^2_{X_{2+}}}}$$

Eine Darstellung des Wertes von δ für jedes Paar der Indices n_η, n_φ findet sich in Abb. 22.

Anhand dieser Darstellung werden die Variablen erstellt: Benachbarte Gebiete, die ein möglichst signifikantes Δ gleichen Vorzeichens aufweisen, werden additiv zusammengefaßt.

Im vorliegenden Beispiel wurden 12 Bereiche ausgewählt, die jeweils eine Variable für ein neuronales Netz lieferten. Die Gebiete, aus denen die Variablen gewonnen wurden, sind in Abb. 22 als Kästchen eingezeichnet und durchnummeriert.

Das Ergebnis eines anschließend trainierten Netzes findet man in Abb. 23.

12 Vergleich der Netze

In diesem Abschnitt sollen die verschiedenen trainierten neuronalen Netze in Bezug auf ihre Reinheit und Effizienz verglichen werden. Diese beiden Größen sind definiert als:

$$\text{Effizienz für die Klasse 1} \quad \epsilon = \frac{N(P(K_1) > \Delta)}{N(K_1)}$$

$$\text{Reinheit für die Klasse 1} \quad R = \frac{N(P(K_1) > \Delta)}{N(P(K_1) > \Delta) + N(P(K_2) > \Delta)}$$

Dabei ist: K_1 (K_2) die Klasse der Ereignisse mit 2+ (0/1) Jets, $N(K_i)$ die Gesamtzahl der Ereignisse der Klasse K_i und $N(P(K_i) > \Delta)$ die Anzahl der Ereignisse aus Klasse K_i, für die der Ausgabewert des Netzes Größer als Δ ist. Δ ist der sogenannte **'probability cut'**, ein Abschneidewert für den Ausgabewert des neuronalen Netzes. Je nachdem, ob der Ausgabewert kleiner oder größer als Δ ist, wird das untersuchte Ereignis der einen oder anderen Ereignis-Klasse zugeordnet.

Die Wahl von Δ muß sich nach den Anforderungen an Reinheit und Effizienz richten.

In Abb. 24 sind Reinheit und Effizienz der drei vorgestellten Netze gegen den Abschneideparameter Δ aufgetragen. Im oberen Teil der Abbildung wird die Entscheidungsqualität bezüglich der Frage „Ist es ein 0/1-Jet Ereignis?“ untersucht, im unteren Teil lautet die Frage „Ist es ein 2+-Jet-Ereignis?“.

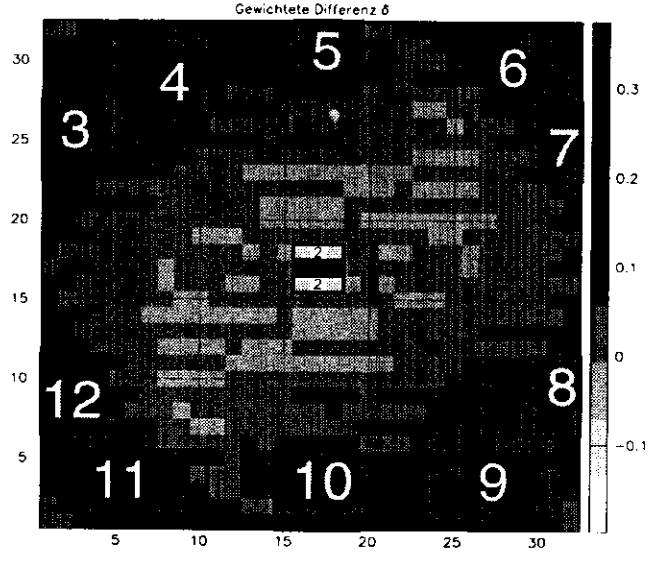


Abbildung 22: Gewichtete Differenz der Mittelwerte. Die als Eingabevariablen verwendeteten Bereiche sind hervorgehoben. Als Achsen dienen die Indices der bei der Fouriertransformation verwendeten Matrix. Die Abszisse entspricht k_η, die Ordinate k_φ.

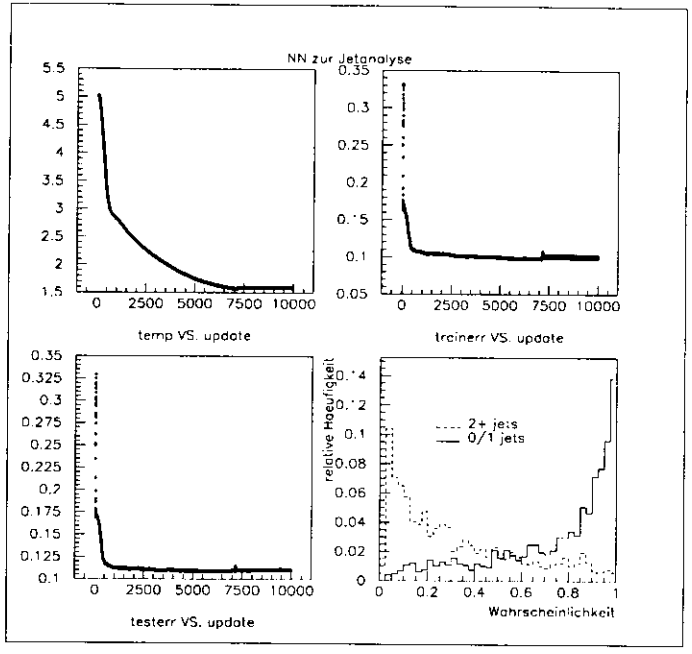


Abbildung 23: Verlauf des Trainings mit 12 Variablen aus der Fouriertransformation. Als Abszisse dient die Nummer der Trainingsiteration (update), aufgetragen sind die Temperatur (temp) und die Fehler für den Trainings- (trainerr) und den Testdatensatz (testerr). Für die Berechnung der Fehler siehe Gl. 35. Rechts unten ist die Verteilung der Ausgabewerte des Netzes dargestellt, die durchgezogene Line wäre im Idealfall ein scharfes Maximum um 1, die gestrichelte ein scharfes Maximum bei 0.

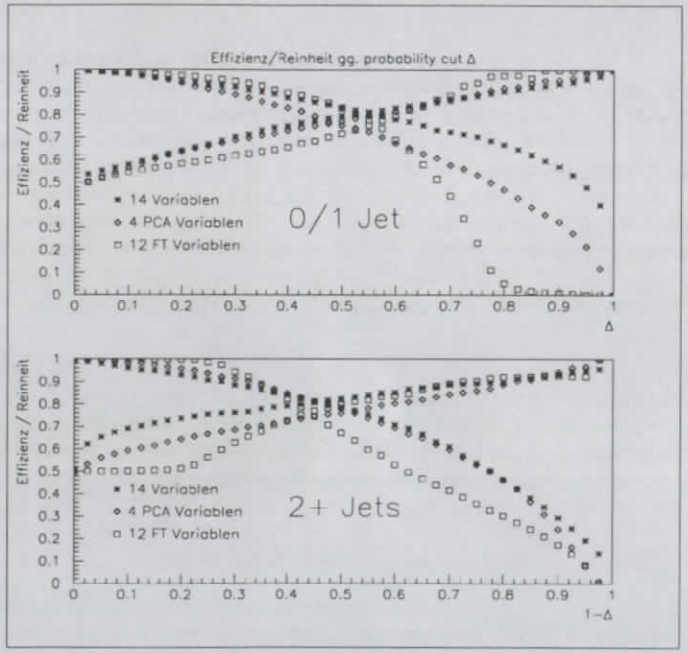


Abbildung 24: Reinheit und Effizienz der untersuchten neuronalen Netze. Die Reinheit steigt jeweils von 50% auf 100%, während die Effektivität von 100% auf 0% fällt.

In der folgenden Tabelle ist die Effizienz der Netze bei der Beantwortung dieser beiden Fragen für je zwei verschiedene Werte der Reinheit festgehalten. Dabei steht *'14 Var'* für das ursprüngliche Netz mit 14 Variablen, *'PCA'* für das Ergebnis der Hauptachsenanalyse und *'FT'* für das mit Variablen aus dem Fourierraum trainierte Netz.

	Effizienz 0/1 Jets			Effizienz 2+ Jets		
Reinheit	14 Var	PCA	FT	14 Var	PCA	FT
90%	68%	52%	42%	44%	38%	34%
95%	51%	34%	19%	17%	20%	16%

Die Leistungsfähigkeit der Netze fällt ab, wenn man zuerst das ursprüngliche Netz mit 14 Eingabevariablen betrachtet, dann das Ergebnis der PCA und zuletzt die Fouriertransformation.

Bei vergleichbarer Reinheit scheint es allerdings leichter zu sein, ein Ereignis mit höchstens einem Jet als ein solches auszuzeichnen, als mit Sicherheit zu sagen, ob in einem Ereignis mindestens zwei Jets vorkommen. Hierbei übertrifft das aus der Hauptachsenanalyse hervorgegangene Netz allerdings das ursprüngliche bei großer Reinheit.

Es ist auch möglich, daß ein weiteres, leistungsfähigeres Netz hätte trainiert werden können, wenn man die Informationen aus dem Fourier-Raum mit denen aus dem Ortsraum verbindet. Im einfachsten Fall würde man hier dann mit $14 + 12 = 26$ Variablen arbeiten.

Kommt ein neuronales Netz zum Einsatz, so kann man einer Darstellung wie Abb. 24 den zu benutzenden Wert für den Abschneideparameter Δ entnehmen. In der Regel ist die Forderung, eine bestimmte Reinheit von typischerweise $R = 0.9$ oder mehr zu erreichen. In dem Diagramm sucht man dann den Wert von Δ heraus, bei dem das entsprechende Netz eine Reinheit des gewünschten Wertes erreicht. Anhand der bei diesem Abschneideparameter erreichten Effektivität kann gleichzeitig überprüft werden, ob das Netz in dieser Form sinnvoll einsetzbar ist.

13 Zusammenfassung

In dieser Arbeit wurde die Möglichkeit untersucht, ein neuronales Netz einzusetzen, um schnell die Anzahl der Jets in Ereignissen tiefunelastischer ep-Streuung bei $Q^2 > 20GeV^2$ zu zählen. Es ist gelungen, ein Netz mit 14 Eingangs-, 5 versteckten und einem Ausgangsknoten zu trainieren, das zwischen zwei Ereignisklassen („0 oder 1 Jet" und „2 oder mehr Jets") unterscheidet, die Eingangsvariablen wurden dabei ausschließlich aus Kalorimeterinformationen konstruiert. Die Trennfähigkeit des Netzes bezüglich der Klasse („0 oder 1 Jet") läßt sich quantitativ durch Reinheit R und Effizienz ϵ ausdrücken. Für Reinheiten von 90% und 95% ergibt sich die Effizienz zu ($\epsilon = 100\%$ würde vollständige Übereinstimmung mit dem als Referenz benutzten Jet-Algorithmus kt bedeuten):

$$\begin{array}{lll} R = 90\% & & R = 95\% \\ \epsilon = 68\% & & \epsilon = 51\% \end{array}$$

Die Methode der Hauptachsenanalyse wurde angewandt, um die Anzahl der benötigten Eingangsvariablen zu reduzieren und die wichtigsten Variablen zu identifizieren. Ein neuronales Netz, das mit vier transformierten Variablen trainiert wurde, erlangte bei einer Reinheit von 90% eine Effektivität von

$$\epsilon = 52\%$$

Weiterhin wurde die Fourier-Transformierte der zweidimensionalen E_t-Verteilung im Kalorimeter benutzt, um die Anzahl von Jets mit Hilfe eines neuronalen Netzes zu beurteilen, welches fouriertransformierte Variable benutzt. Geeignete Variablen im Fourierraum wurden mit Hilfe eines statistischen Verfahrens ermittelt. Der Einsatz der Fouriertransformation war erfolgreich, in Verbindung mit dem systematischen Ansatz zur Variablensuche konnte ein neuronales Netz trainiert werden, das die zwei Ereignisklassen trennen konnte. Die Leistungsfähigkeit liegt aber unter der des ursprünglichen Netzes. Sie kann möglicherweise durch Hinzunahme weiterer, evtl. physikalisch motivierter Variablen verbessert werden.

Abbildungsverzeichnis

A How to use the dEXTRa configuration file

In order to run dEXTRa, you have to set up a configuration file named `dextra.cnf`. A sample file is included at the end of this text.

Please be careful

- to use a **maximum of 100 input cells**, as all arrays in dEXTRa are initialised to this value. Entering more input cells is possible but will have undefined results.
- to make the number of patterns per update of training (NoU) a divisor of the total number of patterns in the learn file.

The networks trainable by dEXTRa are feed-forward, layered networks with one hidden layer and sigmoid transfer functions. Details can be found in [9].

The following parameters are used:

Ninp: Number of nodes in input layer

Nhid: Number of nodes in hidden layer. This is equivalent to the number of hyperplanes by which the data in input space may be subdivided for classification. Note: Entering 0 nodes will result in a simple perceptron with no hidden layer.

Nout: Number of nodes in output layer.

NoU: Number of patters per update of training. Using a large number will reduce the influence of single bits of data, as the gradient is averaged over many patterns. This value is usually set to 1, for an update after each pattern, or to the total number of patterns.

IoF: Sigmoid function. Enter 1 for $\tanh(x)$ as transfer function or 2 for $0.5 * (1 + \tanh(2x))$.

eta: Learning parameter. Eta is directly proportional to the change made to the weights on updating. A small value increases learning time, too large a value increases learning time as well, as the process tends to overshoot the globel minimum of the energy function.

In this model eta=1+gamma+tanh(1/T)**2

mom: Momentum factor. The momentum factor gives the importance of the previous value of the weight. It helps to keep weights from oscillating.

T: Temperature. With typical starting values of 5, the temperature is introduced to smooth out the error/energy Function of the network, thus suppressing local minima and increasing the probability of finding the global minimum.

WoiW: Width of initial weights. This value determines the range in which the weights will be initialised at random, namely between -WoiW and +WoiW.

gamma: Offset for eta (t). This a measure of the minimum increment, usually set to about 0.1.

old DUMP Matrices: If you wish to further the training of an existing network, write the filename of the old network file here.

new DUMP Matrices: This is where the new network file goes

learn INPUT: The file containing the input data. Please make sure to have just one value per line.

learn TARGET: These are the values you would like the netword to come up with after computation.

learn ERROR: The Sum Square error during training goes here. It is divided by the number of patterns, but NOT by the number of output nodes.

net TEMPERATURE: The temperature of the net is written in this file.

test INPUT: Store your test input patterns in this file

test TARGET: This is the wished output during testing.

test ERROR: This is where the error made by the network during testing is written.

test NETWORK OUTPUT: This is the output made by the network during testing.

test HIDDEN: When this option is set, the values of the hidden nodes are output during testing.

learn phase ?(y/n): Write here wether you want to train a network or just test it

NoLP: Number of learn-epochs. This is how many times NoU patterns are used for learning. Example: NoU=10, NoLP=2, 20 patterns are stored in the learn INPUT file. =¿ 2 times 10 patterns are input, the changes are computed and applied. The input file is processed only once. (10*2=20).

temperatur free (y/n): This option specifies whether the temperature is a constant or is allowed to change during training.

monitoring of temperature (y/n): specify here whether you want the temperature to be output or not

normalization of weights (y/n): specify here whether the weights are to be normalized or not. If yes, the squared sum of the weights of any single node will be normalized to 1.

test phase ? (y/n): specify here whether you want a test phase or just a learning phase.

calculate error for learn file (y/n): specify here whether you want the error to be calculated.

calculate error for test file (y/n): specify here whether you wnat the error to be written to file during testing.

ErNode: Specify here the number of the node for which the error is to be computed (when using multiple output nodes).

Node Nr / Crit: *undocumented*

use old dump: If you want to go on training an existing network, select 'y'.

write new dump file: Do you want the network to be written to disk?

write hidden values for test file: enter 'y' if you want the output of hidden nodes during testing to be written to a file.

Note: Should dEXTRa be changed, it might be of advantage to include code that has the status of the net written to file periodically during training, in case program execution is ended abnormally!

A sample dEXTRa configuration file:

```
############## DEXTRA - Configuration - File ###############

Comments ....


Begin
     # nodes in input layer [Ninp] -= input variables----NET-
2
     # nodes in hidden layer [Nhid] = number of hyperplanes
2
     # nodes in output layer [Nout]
1
     number of patterns per update of training [NoU]
4
     sigmoid function / 1=tanh x / 2=0.5*(1+tanh 2x) [IoF]
1
     learning parameter 1/N [eta]
2.5E-4
     momentum factor [mom]
0.5
     network temperatur [T]
5.0
     width of inital weights [WoiW]
1.0
     offset for eta(t) for t->0 [gamma]
0.1
     old DUMP Matrices [1] --------------------------FILES---
```

```
old.dmp
     new DUMP Matrices [2]
new.dmp
     learn INPUT [3]
learn.inp
     learn TARGET [4]
learn.target
     learn ERROR [5]
train.err
     net TEMPERATURE [16]
train.tmp
     test INPUT [6]
test.inp
     test TARGET [7]
test.target
     test ERROR [8]
test.err
     test NETWORK OUTPUT [9]
testoutput.net
     test HIDDEN [19]
n
     learn phase ? (y/n) [10] -------------------------LEARN-
y
     number of learn-epochs [NoLP]
500
     temperatur free (y/n) [11]
y
     monitoring of temperature (y/n) [17]
y
     normalization of weights (y/n) [21]
y
     test phase ? (y/n) [20] ----------------------------TEST-
y
     calculate error for learn file (y/n) [12] ---------ERROR-
y
     calculate error for test file (y/n) [13]
y
     Error calculation for which output node [ErNode]
```



```
1
     Node Nr / Crit ( Node Nr = 0 <> no Crit ) [ErCrit]
0 0
     use old dump ( y/n ) [14] -------------------------DUMP-
y
     write new dump file ( y/n ) [15]
y
     write hidden values for test file (y/n) [18] -----HIDDEN-
n
######################## End ########################
```

Literatur

[1] S. UDLUFT: *Untersuchungen zu Neuronalen Netzen als Vertextrigger im H1-Experiment bei HERA.* Diplomarbeit, Fakultät für Physik der Ludwig-Maxmimilians-Universität München, 1996.

[2] D. WESTNER: *Ein neuronaler Netzwerktrigger für die Produktion von J/Ψ-Teilchen in ep-Streuung ($J/\Psi \to e^+e^-$).* Diplomarbeit, Fakultät für Physik der Ludwig-Maxmimilians-Universität München, 1996.

[3] P. MAZZANTI, R. ODORICO: *Bottom jet recognition by neural networks and statistical discriminants.* Zeitschrift für Physik C, 59:273–282, 1993.

[4] R. SINKUS, H. ABRAMOWICZ: *Electron Identification with Neural Networks at ZEUS.* ZEUS Note 93-117, 1993.

[5] J. HERTZ, A. KROGH, R. G. PALMER: *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.

[6] R. ROJAS: *Theorie der neuronalen Netze.* Springer, 1993.

[7] PRESS ET AL.: *Numerical Recipies.* Cambridge University Press.

[8] U. FRICKE: *Identifikation von neutralen Pionen im ZEUS-Detektor durch feed forward-neuronale Netzwerke.* Interner Bericht DESY F35D-96-09, 1996.

[9] R. SINKUS: *A novel approach to error function minimization for feed-forward neural networks.* DESY 94-182, 1994.

[10] ZEUS COLLABORATION: *A Search for Excited Fermions in e^+p Collisions at HERA.* DESY 97-112, 1997.

[11] *The ZEUS Detector, Status Report 1993, DESY 1993.*

[12] ZEUS COLLABORATION: *A Measurement of $\sigma_{tot}(\gamma p)$ at $\sqrt{s}$ =210 GeV.* DESY 92-127, 1992.

[13] O. DEPPE: *Messung des hadronischen Energieflusses in der tiefunelastischen e-p-Streuung.* Diplomarbeit, Universität Hamburg, 1994.

[14] P. DE JONG: *Status of the Uranium Calorimeter Reconstruction Software.* ZEUS-Note 92-019, 1992.

[15] M. DE KAMPS: *Changes and extensions of the calorimeter reconstruction programme.* ZEUS-Note 94-014, 1994.

[16] PROGRAMMING TECHNIQUES GROUP: *ADAMO Version 3.3 Users guide.* ECP Division, CERN, October 1993.

[17] E. LOHRMANN: *Einführung in die Physik der Elementarteilchen.* Teubner Verlag, 1983.

[18] S. BENTVELSON, J. ENGELEN AND P. KOOIJMAN: *Reconstruction of (x, Q^2) and extraction of structure functions in neutral current scattering at HERA.* In: *Physics at HERA*, Band 1, Seiten 23–41. DESY, 1991.

[19] M.H. SEYMOUR: *Jets in QCD.* In: RAJA, RAJENDRAN und JOHN YOH. (Herausgeber): *Proton-Antiproton Collider Physics,Proceedings, 10th topical workshop, Batavia, USA*, Seite 734. AIP, 1995.

[20] E. LOHRMANN: *Hochenergiephysik.* Teubner, 1992.

[21] S. CATANI ET AL: *Longitudinally-invariant $k_\perp$-clustering algorithms for hadron-hadron collisions.* Nuclear Physics B, 406:187–224, 1993. WWW: `http://surya11.cern.ch/users/seymour/ktclus/`.

[22] T. TREFZGER: *Messung der Kopplungskonstanten α_S der starken Wechselwirkung aus Jetraten in der tiefunelastischen ep-Streuung bei HERA.* Doktorarbeit, Albert-Ludwigs-Universität Freiburg i. Br., März 1996.

[23] *DJANGO6 version 2.4 - A Monte Carlo Generator for Deep Inelastic Lepton Proton Scattering Including QED and QCD Rediative Effects.*

[24] S. MAGILL: *A Measurement of the Inclusive Jet Production Cross Section in DIS at HERA.* ZEUS Note 94-154, 1994.

[25] ZEUS COLLABORATION: *Jet Production in High Q^2 Deep-Inelastic ep Scattering at HERA.* DESY 95-016, 1995.

[26] TASSO COLLABORATION: *Evidence for planar events in e^+e^- annihilation at high energies.* Physics Letters, 86B:243–249, 1979.

[27] PLUTO COLLABORATION: *Evidence for gluon Bremsstrahlung in e^+e^- annihilation at high energies.* Physics Letters, 86B:418–425, 1979.

[28] H. WIND: *Principal Component Analysis and its Applications to Track Finding.* R.K. Bock: Formulae and Methods in Experimental Data Evaluation, Vol. 3:K1–K16, 1984.

[29] R. SINKUS: *Measurement of the Proton Structure Function F_2 from the 1994 HERA Data Using a Neural Network for the Identification of the Scattered Lepton.* Dissertation, Universität Hamburg, 1996.

[30] M. LIEBE: *Filteralgorithmen zur Bestimmung von y_{JB} in der tiefunelastischen Streuung.* Diplomarbeit, Universität Hamburg, Oktober 1995.

Erklärung

Hiermit versichere ich, daß ich die vorliegende Arbeit unter Angabe aller verwendeten Quellen und Hilfsmittel selbständig angefertigt habe.

Hamburg, im September 1997

(Michael Sievers)