Neural Description Language (NDL)

El **Neural Description Language (NDL)** és un llenguatge dissenyat per descriure, configurar i gestionar xarxes neuronals. Aquest llenguatge es compon de diversos tipus de fitxers que permeten separar l'estructura, el contingut, la configuració d'entrenament i els punts de control.

Tipus de fitxers del Llenguatge NDL

1. .ndl - Neural Description Language (Estructura)

El format *'.ndl'* s'utilitza per definir l'arquitectura d'una xarxa neuronal, incloent-hi el nombre de capes, el nombre de neurones per capa i les connexions entre elles.

Estructura Genera

Un fitxer .ndl és un document JSON que conté els següents camps:

- *name*: (Obligatori) Nom de la xarxa.
- *description*: (Opcional) Descripció de la xarxa.
- *layers*: (Obligatori) Llista que especifica el nombre de neurones en cada capa.
- *links*: (Obligatori) Llista de connexions entre neurones.

Exemple:

```
{
  "name": "my_network",
  "description": "Descripció opcional",
  "layers": [40, 10, 10, 5],
  "links": [
    {"from": "L:0, N:n", "to": "L:1, N:n"},
    {"from": "L:1, N:n", "to": "L:2, N:n"},
    {"from": "L:y, N:n", "to": "L:y+1, N:n"},
    {"from": "L:4, N:3", "to": "L:2, N:7"},
    {"from": "L:2, N:3", "to": "L:4, N:n"}
  ]
}
```

Descripció dels Camps

1. **name** (obligatori)
   Tipus: Cadena de text
   Descripció: Nom identificatiu de la xarxa neuronal.

2. **description** (opcional)
   Tipus: Cadena de text
   Descripció: Descripció opcional que proporciona informació addicional sobre la xarxa.
3. **layers** (obligatori)
   Tipus: Llista d'enters
   Descripció: Cada element de la llista representa el nombre de neurones en la capa corresponent, començant per la capa d'entrada.
   Nota: La longitud de la llista determina el nombre total de capes de la xarxa. Per exemple, *"layers": [40, 10, 10, 5]* indica una xarxa amb quatre capes: una capa d'entrada amb 40 neurones, dues capes ocultes amb 10 neurones cadascuna i una capa de sortida amb 5 neurones.
4. **links** (obligatori)
   Tipus: Llista d'objectes
   Descripció: Cada objecte de la llista defineix una connexió entre neurones, amb els camps:
   - from: Especifica la(es) neurona(es) d'origen.
   - to: Especifica la(es) neurona(es) de destinació.

Format per a from i to: "L:x, N:y":

- *L:x*: Indica la capa (x).
  - x pot ser un número específic o la lletra y per referir-se a totes les capes.
- *N:y*: Indica la neurona (y) dins de la capa.
  - y pot ser un número específic o la lletra n per referir-se a totes les neurones.

Exemples:

- "L:0, N:n": Totes les neurones de la capa 0.
- "L:y, N:n": Totes les neurones de totes les capes.
- "L:2, N:3": La quarta neurona de la tercera capa.
- From: "L:2, N:3", "L:4" és equivalent a "L:2, N:3", "L:4, N:n".
- "L:y", "L:y+1" és equivalent a "L:y, N:n", "L:y+1, N:n".

2. .ncl - Neural Content Language (Contingut)

Estructura General

Un fitxer .ncl és un document JSON que conté els següents camps:

- **specification**: (Obligatori) Ruta o identificador del fitxer .ndl corresponent
- **NFs**: (Opcional) Llista de funcions d'activació assignades a neurones o capes. Per defecte la funció escollida serà "ReLU"

- **weights**: (Opcional) Llista de pesos per a les connexions sinàptiques. Per defecte els pesos de *from* a *to* es crearan amb valors al·leatoris Xavier/Gorot
- **biases**: (Opcional) Llista de valors de biaix per a les neurones. Per defecte els biaixos de from a *to* es crearan amb valor '0.0'

Exemple:

```
{
  "specification": "github.com/jibort/.../a.ndl",
  "NFs": [
    { "to": "L:1, N:n", "functions": ["ReLU", "Tanh"] },
    { "to": "L:1, N:3", "functions":  ["Sigmoid"] },
    { "to": "L:y, N:1", "functions":  ["Tanh", "ReLU"] },
    { "to": "L:1", "functions":  ["Sigmoid", "ReLU", "Linear"] }
  ],
  "weights": [
    { "from": "L:0", "to": "L:1", "value": "-0.332"},
    { "from": "L:2", "to": "L:4, N:3", "value": "0.52"},
    { "from": "L:y, N:0", "to": "L:y+1, N:0", "value": "0.1*"},
    { "from": "L:2", "value": "RXG"},
    { "from": "L:2", "value": ["RNG"]}
  ],
  "biases": [
    {"from": "L:y", "value": "-0.59999"},
    {"from": "L:2", "value": "0.59999"},
    {"from": "L:y", "to": "N:3", value: "-0.31415"}
  ]
}
```

Descripció dels Camps

1. **specification**
    - Tipus: Cadena de text
    - Descripció: Identifica el fitxer .ndl que defineix l'estructura de la xarxa a la qual es refereixen els valors proporcionats.
2. **NFs** (Neural Functions)
    - Tipus: Llista d'objectes
    - Descripció: Assigna funcions d'activació a neurones o capes específiques. Cada objecte conté una clau que especifica la neurona o capa i un valor que és una llista de funcions d'activació.
    - Format:

      ```
      "NFs": [
        {to: "L:x, N:y", "functions": ["Funció1", "Funció2"]},
        ...
      ]
      ```

      On:

        - *x* indica la capa x. Pot ser un número específic o la lletra ‘y’ per referir-se a totes les capes.

- *N:y*: Indica la neurona y dins de la capa. Pot ser un número específic o la lletra ‘n’ per referir-se a totes les neurones.
- Es segueixen tots els criteris descrits prèviament pels camps ‘from’ i ‘to’.
- ´La llista de cadenes de caràcters *functions* estableix totes les funcions neuronals que s’executaran per ordre un cop s’hagi obtingut el resultat del càlcul sobre pesos i biases.

Exemples:

- {"to": "L:1, N:n", "functions": ["ReLU", "Tanh"]}: Assigna les funcions d'activació *ReLU* i *Tanh* a totes les neurones de la capa 1.
- {"to": "L:y, N:1", "functions": ["Tanh", "ReLU"]}: Assigna les funcions d'activació *Tanh* i *ReLU* a la primera neurona de totes les capes.

- *weights* (Pesos)
  - Tipus: Llista d'objectes
  - Descripció: Defineix els pesos de les connexions sinàptiques entre neurones. Cada objecte especifica la neurona d'origen, la neurona de destinació i el valor del pes.
  - Format:

```
"weights": [
  {"from": "L:x, N:y", "to": "L:z, N:w", "value": "pes"},
  ...
]
```

On:

- from: Neurona d'origen, especificada com "L:x, N:y".
- to: Neurona de destinació, especificada de la mateixa manera.
- value: Valor del pes, que pot ser un número o una cadena que indica un mètode d'inicialització aleatòria (per exemple, "RXG" per a Xavier/Glorot) o valors especials com «+inf», «-inf», «NaN», «nul», ...

Exemples:

- {"from": "L:0", "to": "L:1", "value": "-0.332"}: Assigna un pes de -0.332 a totes les connexions de la capa 0 a la capa 1.
- {"from": "L:2", "value": "RXG"}: Inicialitza aleatòriament els pesos de totes les connexions que surten de la capa 2 segons el mètode Xavier/Glorot.
- "biases" (Biaixos)
  - Tipus: Llista d'objectes

- Descripció: Especifica els valors de biaix per a neurones individuals o grups de neurones. Cada objecte indica la capa i/o neurona i el valor del biaix.
- Format:

```
"biases": [
  { neurons: "L:x, N:y", "bias": "value" },
  ...
]
```

On:

- *'L:x'*: Indica la capa x.
- 'N:y': Indica la neurona y dins de la capa.
- '*biaix*': Valor del biaix assignat.

Exemples:

- {neurons: "L:y", value: "-0.59999"}: Assigna un biaix de -0.59999 a totes les neurones de totes les capes.
- {neurons: "L:2", value: "0.59999"}: Assigna un biaix de 0.59999 a totes les neurones de la capa 2.
- {neurons: "L:2, N:6", value: "0.59999"}: Assigna un biaix de 0.59999 a la neurona 6 de la capa 2.

3. .ntl - Neural Training Configuration (Configuració d'Entrenament)

Aquest fitxer defineix els paràmetres per configurar l'entrenament de la xarxa.

Exemple:

```
{
  "learningRate": 0.001,
  "batchSize": 32,
  "epochs": 100,
  "optimizer": "Adam",
  "lossFunction": "MSE"
}
```

4. .nkl - Neural Checkpoint (Punt de Control)

Aquest fitxer desa l'estat complet de la xarxa.

Exemple:

```
{
  "epoch": 42,
  "weights": [
     [0.52, 0.21],
     [0.13, 0.79],
     [0.34, 0.42] ]
   ],
   "biases": [
      [0.12, 0.22, 0.32],
```

```
      [0.45, 0.53]
  ],
  "functions": [
      ["ReLU", "Tanh", ...],
      ["Sigmoid", "ReLU", "Linear"],
      ...
   ],
}
```

Exemple d’ús

```
// Crear una xarxa des de l'estructura
network, err := NewNetworkCPU("my_network.ndl")
if err != nil {
    log.Fatal(err)
}

// Carregar els valors inicials
err = network.LoadContent("my_network_values.ncl")
if err != nil {
    log.Fatal(err)
}

// Entrenar la xarxa
trainer := NewTrainer("my_training_config.ntr")
trainer.Train(network)

// Desar el punt de control
err = network.SaveCheckpoint("my_network_checkpoint.chk")
if err != nil {
    log.Fatal(err)
}
```