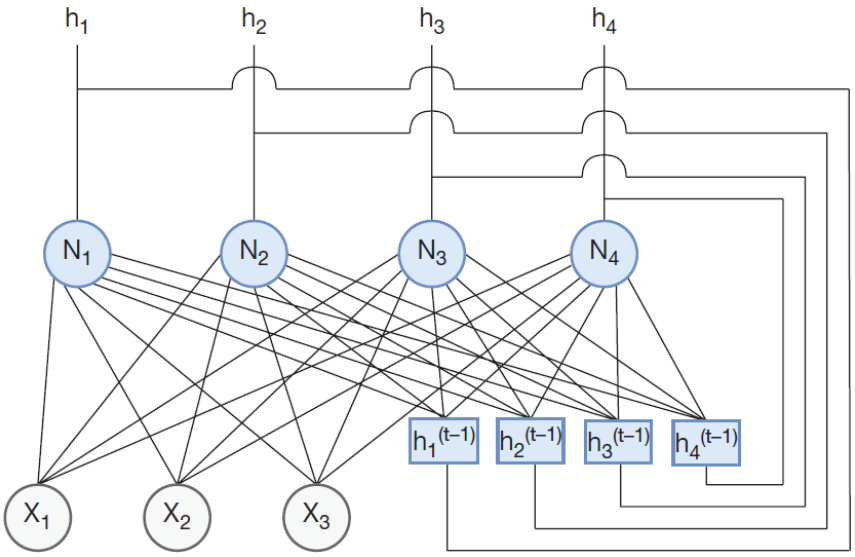
7. Rețele neuronale recurente

Această arhitectură este utilă atunci când se fac predicții bazate pe date secvențiale și, mai ales, pentru secvențe de lungimi variabile.

O formă simplă de rețea neuronală recurentă poate fi creată prin conectarea ieșirilor unui strat complet conectat la intrările aceluiași strat:

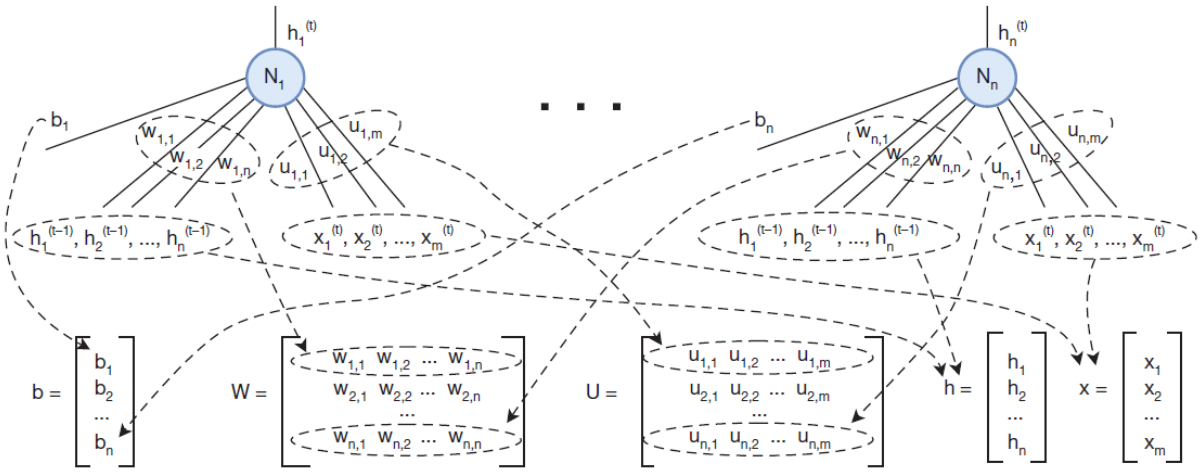


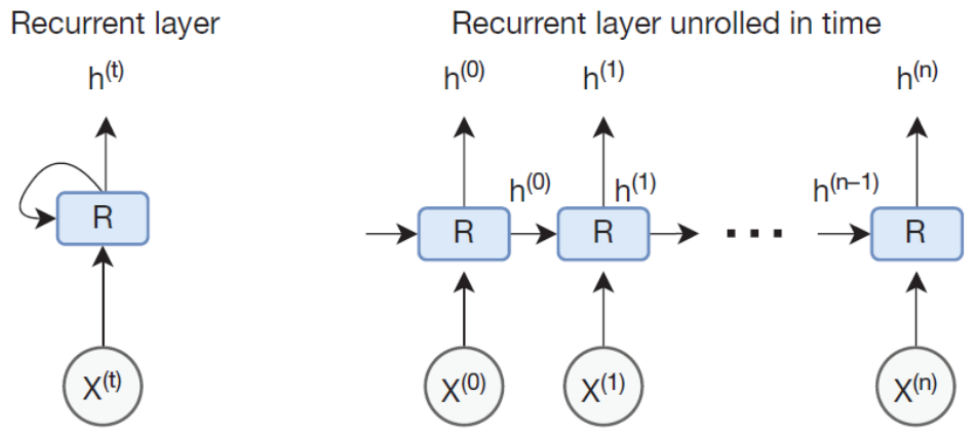
Reprezentarea matematică a unui strat complet conectat:

$$\mathbf{y} = \tanh\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right)$$

Reprezentarea matematică a unui strat recurent:

$$\mathbf{h}^{(t)} = \tanh\left(\mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} + \mathbf{b}\right)$$





Word embeddings

```
import numpy as np
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
```

```
docs = ['Well done!',
        'Good work',
        'Great effort',
        'nice work',
        'Excellent!',
        'Weak',
        'Poor effort!',
        'not good',
        'poor work',
        'Could have done better.']

labels = np.array([1, 1, 1, 1, 1, 0, 0, 0, 0, 0])

vocab_size = 50
encoded_docs = [one_hot(d, vocab_size) for d in docs]
print(encoded_docs)
```

```
[[17, 17], [24, 5], [14, 45], [22, 5], [40], [25], [33, 45], [18, 24],
[33, 5], [40, 48, 17, 14]]
```

```
max_length = 4
padded_docs = pad_sequences(encoded_docs, maxlen=max_length, padding='post')
print(padded_docs)
```

```
[[17 17  0  0]
 [24  5  0  0]
 [14 45  0  0]
 [22  5  0  0]
 [40  0  0  0]
 [25  0  0  0]
 [33 45  0  0]
 [18 24  0  0]
 [33  5  0  0]
 [40 48 17 14]]
```

```
model = Sequential()
model.add(Embedding(vocab_size, 8, input_length=max_length))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy',
              metrics=['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 4, 8)              400
_________________________________________________________________
flatten (Flatten)            (None, 32)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 33
=================================================================
Total params: 433
Trainable params: 433
Non-trainable params: 0
_________________________________________________________________
```

```
model.fit(padded_docs, labels, epochs=50, verbose=1)

loss, accuracy = model.evaluate(padded_docs, labels, verbose=0)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

```
Accuracy: 80.00%
```