**RIASSUNTO SEMINARI DI RNDL 20/21**

**GAIA**

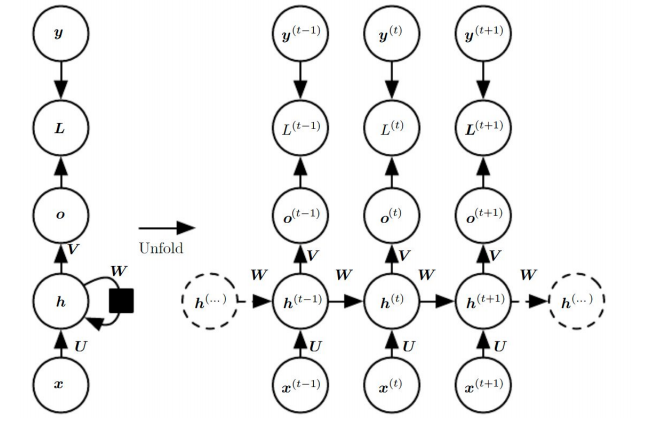
Gaia è un satellite che calcola la posizione delle stelle vicine al pianeta terra grazie a delle componenti in grado di catturare le onde elettromagnetiche che le stelle emettono. Queste onde sono state utilizzando per costruire dei plot che rappresento i dati ricevuti dal satellite.

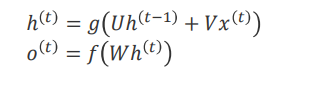
A causa della grande mole di plot che il satellite raccoglie è stato deciso di utilizzare una CNN, allenata per scovare le anomalie presenti in questi plot.

**Recurrent Neural Networks**

La grande differenza con le altre reti neurali è che le RNN si basano su **dati sequenziali**, ossia dati in cui l’ordine è importante(Ex: testo,audio). Per questo motivo le RNN introducono il concetto di ciclo e di temporalità.

A ogni istante T la rete utilizza la computazione all’istante t-1 per calcolare l’input. Per questo motivo c’è una forte dipendenza tra il calcolo dell’input x nell’istante T con le computazioni precedenti.





dove :

* U è la matrice dei pesi del livello input-to-hidden
* W è la matrice dei pesi dei livelli hidden-to-hidden
* V è la matrice dei pesi del livello output-to-hidden

**Vanilla RNN** è una rete di questo genere che utilizza come:

g la funziona tanh

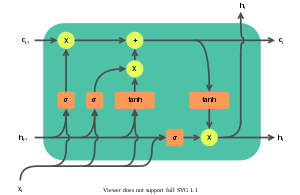
f la funziona softmax

Questo tipo di reti utilizza la **Backpropagation through time**. Per ogni sequenza è necessario un intervallo di tempo, ogni esempio richiede un istante t in cui viene calcolata la loss. Al termine della sequenza, attraverso la discesa del gradiente, vengono aggiornati i mesi tenendo conto della somma di tutte le loss calcolate.

**PROBLEMA:** Soffrono con dati sequenziali molto lunghi in quanto gli input più giovani sono influenzati pochissimo da quelli più vecchi.

Esistono 2 importanti modelli:

**LSTM** (Long short term memory), risolvono il problema appena citato, in quanto sono composti cosi:



dove c è la catena che trasporta le informazioni delle computazioni precedenti. Ma questo modello è in grado di risolvere il problema aggiungendo e rimuovendo informazioni/input che hanno poco valore per il task.

**BLSTM** la differenza sostanziale è che sono bidirezionali quindi le computazioni possono sia dipendere dai dati precedenti che da quelli successivi.

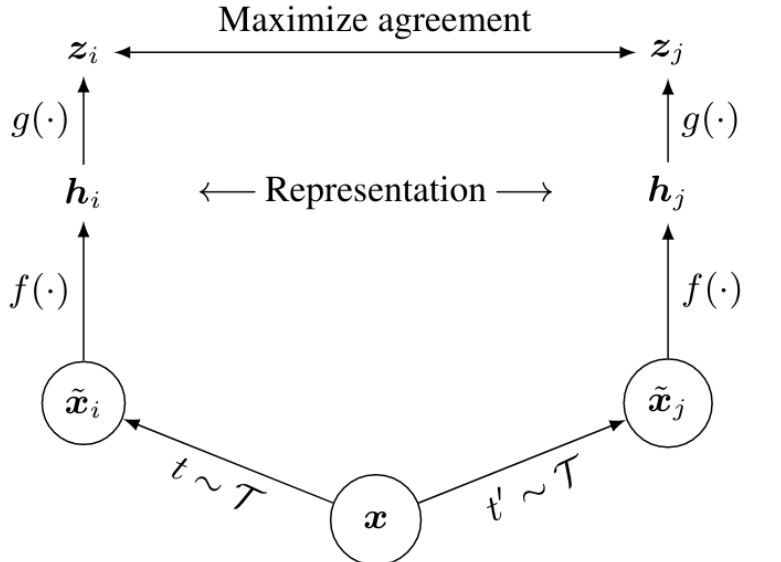
**SEMI SUPERVISED LEARNING**

Per risolvere il problemi de SSL esistono varie tecniche tra cui:

**Smoothness assumption,** fare assunzioni sulle classi degli esempi in base alla loro distanza e alla densità della regione in cui si trova. ( Simile alla clusterizzazione). Viene seguita dal **pseudo labelling** che è semplicemente assegnare le etichette in base alle assunzioni, utilizzando la funzione di attivazione come confidenza.

**SELF SUPERVISED TRAINING.**

Si utilizzano le **reti siamesi.**

****

Supponendo che x sia un’immagine:

Si prende x e si effettuano due trasformazioni che vengono rispettivamente passate come input alle due reti, che si allenano cercando di ottenere lo stesso risultato.

**LEARNING SPARSE NEURAL TOPOLOGIES**

Consiste semplicemente nell’analizzare due t ipi di compressione delle reti neurali il cui obiettivo è quello di rimuovere alcuni archi tra i neuroni senza compromettere le performance del modello( è uguale a mettere il valore dei pesi uguali a 0).

**Quantizzazione:** in cui i pesi vengono sempre trasformati in interi, per cui i numeri prossimi allo zero vengono direttamente convertiti in 0.

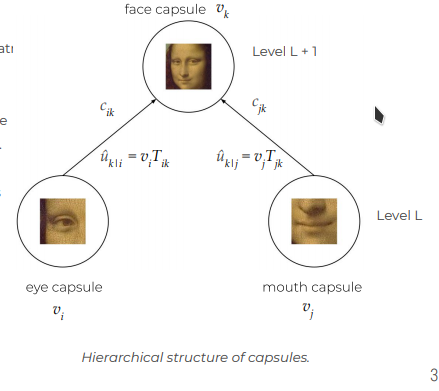
**Sparsificazione:** Consiste nell’effettuare il pruning degli archi. Lo si fà attraverso un algoritmo che prevede un iperparametro T (Threshold) e mentre si allena la rete i neuroni con peso <T vengono settati a 0.

**Capsule Network**

Nascono perchè il Max Pooling nelle CNN non sono in grado di catturare le informazioni spaziali e le relazioni tra le entità contenute nelle immagini. Per questo motivo nascono le Capsule Network, ossia reti neurali organizzate in **Capsule** e in **Livelli**.

**Capsula** è un gruppo di neuroni che si occupano di calcolare la probabilità che una determinata entità, di cui si occupa una capsula, sia presente nell’immagine e la sua posa( orientamento nello spazio).

L'idea principale alla base di questo è creare **equivarianza** tra le capsule. Ciò significa che spostare un elemento in un'immagine cambierà anche la sua rappresentazione vettoriale nelle capsule, ma non la probabilità che esista. Dopo che le capsule di livello inferiore hanno rilevato le funzionalità, queste informazioni vengono inviate verso le capsule di livello superiore che si adattano bene.



**I pesi decidono a quale capsula di livello superiore la capsula attuale invierà il suo output. Questo viene fatto attraverso un processo di routing dinamico.**

**SAR CNN**

Utilizzare le CNN per elaborare le immagini ottenute dai radar SAR.

Ovviamente queste immagine sono molto rumorose per cui l’intuizione è quella di utilizzare **Denoising AutoEncoders**. Questi modelli devono essere allenati per rimuovere il rumore presente nell’immagine per cui l’obiettivo è quello di ottenere come output del modello il rumore presente nell’immagine in maniera tale da sottrarlo all’immagine iniziale e ottenere l’immagine senza rumore.

Siccome il satellite cattura la stessa immagine più volte nel tempo è stato deciso di implementare un modello chiamato **SAR CNN.**

**NON CI HO CAPITO UN CAZZO! XD**

**ADVERSARIAL EXAMPLES**

Consiste sostanzialmente nel generare esempi in grado di confondere la rete neurale in maniera tale da verificare quanto hanno veramente imparato. Nel caso di classificatore di immagini questo tipo di problema si riduce al semplice fatto di aggiungere una quantità di rumore impercettibile all’occhio umano e verificare se la rete neurale predice ancora correttamente. Per realizzare questo tipo di esempi si usano due algoritmi:

* **L-BFGS** che risolve un problema di ottimizzazione cercando appunto il minimo rumore in grado di confondere la rete neurale
* **FAST GRADIENT SIGN METHOD,** ossia il rumore generato dipende dal segno del gradiente e da un Epsilon indicato dall’utente. (MOLTO MEGLIO)

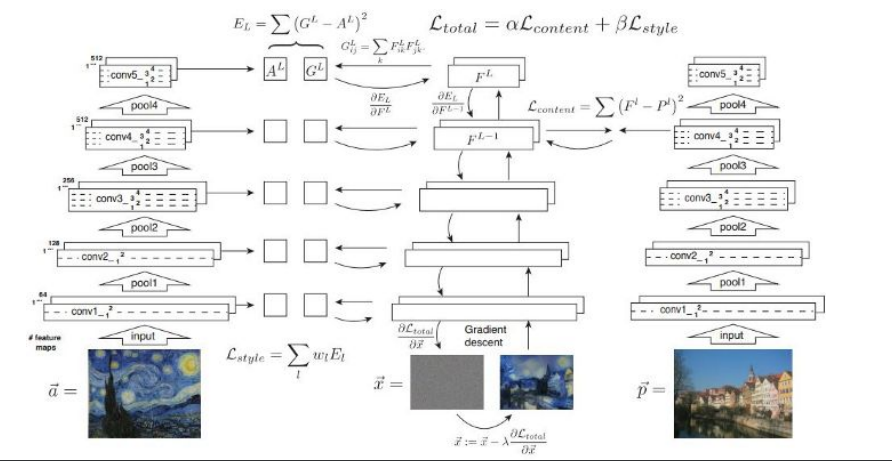
**ARTISTIC STYLE**

Utilizza 2 CNN: una per riconoscere il contenuto del quadro e una che riconosce lo stile.

Ogni CNN ha una loss, che sono rispettivamente Content Loss e Style Loss. In fase di apprendimento l’obiettivo è quello di minimizzare la somma delle due loss.Entrambi le CNN prendono in input due immagini: una normale e una generata dal rumore bianco.

La prima CNN, che si occupa del contenuto usa la similarità e la discesa del gradiente

La seconda CNN: utilizza la GRAM MATRIX, dove ogni elemento è il prodotto interno dei pixel. La loss corrisponde alla distanza quadratica media tra le due immagini.



**UNSUPERVISED LEARNING WITH DCGAN**

Sono state utilizzate immagini naturali per fare gli esperimenti.

Questa architettura è costituita da una GAN, in cui sia il generatore che il discriminatore sono una CNN, in cui:

* il livello di pooling è sostituto con con uno stride convolution ( come un pool ma con stride > 1)
* Batch Normalization, i dati vengono sempre normalizzati
* Vengono rimossi gli ultimi livelli Fully connected perchè non bisogna fare classificazione
* Viene usata la ReLU invece della tanH

Questo modello viene utilizzato per inizializzare i pesi e poi effettuare il transfer learning su un altro task.

**XNAS**

XNAS è un tool per trovare la miglior architettura per un determinato task. Queste metodologie sono costituite da 3 elementi:

1. Search space, definisce quanti e quali livelli utilizzare
2. Search Strategy, ossia l’algoritmo che trova l’architettura migliore
3. Performance Strategy, il metodo usato per valutare le varie soluzioni

XNAS calcola tutte le possibili combinazioni di operazioni che possiamo fare per costruire una rete cercando di escludere man mano quelle pessime. Quindi si basa sull'algoritmo DARTS, aggiungendo la PEA ( Prediction with Expert Advice), ossia un sistema in cui ci sono due attori:

* Gli esperti che forniscono al Decision Maker la propria predizione sulle possibili operazioni
* Decision Maker che analizzando le soluzioni proposte dagli esperti prende la decisione finale.

Utilizza la discesa del gradiente esponenziale per aggiornare i pesi.

**MACHINE TRANSLATION**

Utilizzo di due architetture ricorrenti : **RNNencdec e RNNSearch**

In entrambi i modelli le parole vengono trasformate in numeri (**word embedding**). Esiste una **funzione di attenzione** ( sigmoide) che permette di dar maggior peso alle parole più importanti in una sequenza.

**RNNencdec:**

**Encoder** esegue il word embedding e calcolare un vettore di contesto a lunghezza fissa.

**Decoder** è una LSTM che prende in input il vettore di contesto e utilizza la softmax per calcolare la probabilità delle possibili parole che traducono l’input.

Problema: Con dati sequenziali lunghi perché le parole più giovani dipendono troppo poco dalle parole più vecchie.

**RNNSearch:**

**Encoder** utilizza una RNN Bidirezionale in maniera tale da considerare anche le parole successive, ( Corrispondono a 2 RNN una che và avanti e una che và indietro), producendo un vettore di contesto a lunghezza variabile.

**Decoder** utilizza una GRU(invece di LSTM) e softmax come prima.

**SPARSITY**

Consiste nel ridurre il numero di archi di una rete neurale in maniera tale da ridurre il costo computazionale senza degenerare le performance.

Questa tecnica, come abbiamo già visto, consiste nell’effettuare il pruning delle connessioni.

L’approccio è non strutturato (sparsificazione senza un criterio topologico) e su reti fully connected.

Ci sono in sostanza 3 metodi:

* **Variational DropOut**, ossia durante l’apprendimento ogni connessione che ha il peso minore di un threeshold viene eliminata.
* **L0 - regulation**, semplicemente si utilizza la norma L0 sui pesi, che in generale tende a portare tutti i valori vicino allo zero.
* **Magnitude Pruning**, dove invece della norma si utilizza il valore assoluto ( non ho capito onestamente in che senso ma non mi ci sono manco voluto impegnare)