PikaPikaGenerator

Generatore di Pokémon Spirtes a partire da Descrizioni Testuali

Progetto di Deep Learning

Corso: Deep Learning – Politecnico di Bari

Studente: Pasquale Alessandro Denora

Docente: Prof. Vito Walter Anelli

Sommario

Abstract

Il progetto PikaPikaGenerator rappresenta un'implementazione completa di un sistema di generazione di immagini a partire da descrizioni testuali, in maniera specifica è progettato per creare sprite Pokémon a partire da descrizioni testuali Pokédex.

Il sistema utilizza un'architettura encoder-decoder, con meccanismo di attenction, dimostrando l'applicazione pratica di tecnologie avanzate nel campo del Natural Language Processing e della Computer Vision.

Risultato Principale: Il modello è stato implementato con successo e genera immagini a partire dalla digitazione in input di testo. Sebbene gli sprite che vengono generato differiscano in maniera notevole dalle immagini reali del dataset, in termini di qualità visiva, riconoscibilità e fedeltà all'immagine originale, il sistema ha appreso le basi della mappatura testo-immagine e produce output visivi, che generalmente sono visivamente interpretabili.

Successo tecnico: L'implementazione completa dell'architettura, dal pre-processing alla demo interattiva, rappresenta un buon punto di partenza per eventuali implementazioni future.

Data Pre-processing

2.1 Dataset e preparazione dei dati

Il progetto utilizza 'The Complete Pokèdex Dataset', una collezione di 898 Pokémon con descrizioni testuali e corrispondenti sprite [1].

Questo dataset fornisce una base solida per l'apprendimento con descrizioni ricche e immagini standard che rappresentano un base di conoscenza appropriata per il nostro obiettivo.

L'analisi esplorativa del dataset ha rivelato caratteristiche interessanti:

- Descrizioni con lunghezza media di 123 caratteri
- Sprites mantengono una consistenza visiva che facilita l'apprendimento

Tuttavia la complessità intrinseca della relazione testo-immagine nel dominio Pokémon si è rivelata significativa.

2.2 Strategia di Pre-processing

Il pre-processing del testo ha implementato una pipeline robusta basata su BERT-mini tokenizer, garantendo una rappresentazione semantica ricca del contenuto testuale.

La gestione di encoding multipli e la validazione dei dati hanno assicurato la qualità degli input di training.

Le sequenze sono state normalizzate a 128 token, bilanciando l'informazione semantica e l'efficienza computazionale.

Per quanto riguarda le immagini, una sfida importante è stata rappresentata dalla gestione della trasparenza. La conversione da RGBA a RGB, con backround bianco ha standardizzato gli input, mentre la normalizzazione pixel e le trasformazioni di augmentation hanno preparato i dati per il training generativo.

Tuttavia, anche considerati gli accorgimenti presi in considerazione, la complessità visiva degli sprites Pokémon rimane una sfida per la generazione fedele delle immagini.

La configurazione tramite file YAML permette facile modifica dei parametri senza cambiare il codice.

2.3 Qualità e Limitazioni dei Dati

Il dataset, sebbene sia ben strutturato, presenta alcune limitazioni intrinseche per quanto riguarda il task generativo. La diversità stilistica degli sprite originali, oltre alle sottili differenze nei dettagli visivi e la complessità delle descrizioni testuali, creano uno spazio di mappatura molto ampio e complesso.

Architettura del Modello

3.1 Design dell'Architettura

L'architettura PikaPikaGenerator implementa un paradigma encoder-decoder con tre componenti principali:

- Test Encoder BERT – based
- Meccanismo multi-head attention
- Generatore CNN residuale

Il design in teoria è solido e segue le best-pratice per sistemi text-to-image.

L'Encoder processa il testo per estrarre rappresentazioni semantiche, il meccanismo di attention allinea informazioni testuali e informazioni visuali, e il generatore traduce le rappresentazioni raffinate in immagini pixel.

3.2 Text Encoder con BERT Pre-Addestrato

Il TextEncoder utilizza BERT-mini come foundation model, sfruttendo i pesi pre-addestrati e applicando il fine-tuning durante il training.

L'implementazione include layer di proiezione che adatta la dimensione di output di BERT alla dimensione hidden desiderata, seguito da LayerNorm e Dropout per stabilizzazione.

Una caratteristica importante è la possibilità di congelare inizialmente i parametri BERT per stabilizzare il training, permettendo poi di scongelarli gradualmente.

Il sistema produce sia sequence_output(per ogni token) che pooled_output, ossia la rappresentazione globale tramite CLS token, fornendo flessibilità per il meccanismo di attention.

3.3 Meccanismo MultiHeadAttention

Il sistema di attenzione implementa 8 attention heads che permettono al modello di catturare diverse tipologie di relazioni semantiche.

L'implementazione calcola quey, key e value attraverso proiezioni lineari e applica il meccanismo di self-attention per raffinare le features testuali.

Il sistema, quindi, restituisce sia l'output processato che gli attention weights, utili per interpretabilità e debugging.

La media degli attention weights tra gli heads fornisce una visualizzazione comprensibile di come il modello interpreta le diverse parti del testo durante la generazione.

3.4 SpriteGenerator CNN Avanzato

Il generatore implementa un'architettura CNN sofisticata che combina features testuali con rumore casuale per produrre immagini.

L'architettura inizia con una proiezione fully-connected che trasforma l'input combinato in una feature map 4x4, poi utilizza sei ResidualBlock sequenziali per upsampling progressivo fino alla dimensione finale.

Ogni ResidualBlock incorpora connessioni skip per migliorare la stabilità del training e facilitare il flusso dei gradienti.

L'upsampling utilizza interpolazione bilineare per transizioni smooth tra le risoluzioni.

La convoluzione finale con attivazione Tanh produce output nell'intervallo [-1,-1], matchando la normalizzazione degli input.

2.5 Integrazione e Funzionalità Complete

La classe PikaPikaGenerator mette insieme tutti i componenti in un sistema end-to-end.

Il metodo forward() implementa il flusso completo: encoding testuale, applicazione attention, generazione immagine.

Il sistema include anche metodi di utilità per generazione diretta da testo generate() e visualizzazione attention (get_attention_visualization()).

L'architettura supporta la generazione controllata tramite il noise input, permettendo di produrre variazioni diverse della stessa descrizione.

L'adaptive pooling finale garantisce dimensioni di output esatte indipendentemente dalle variazioni durante l'upsampling.

Processo di Training

4.1 Strategia di Training End-to-End

Il sistema di training implementato nella classe Trainer gestisce l'ottimizzazione simultanea di tutti i componenti dell'architettura.

L'approccio end-to-end permette al modello di comprendere la mappatura testp-immagine in modo olistico, ottimizzando l'intero blocco pipeline per l'obiettivo finale di generazione.

L'implementazione include support per Automatic Mixed Precision per efficienza computazionale, gradient clipping per stabilità ed early stopping per prevenire overfitting.

Il sistema infatti è progettato per gestire training prolungati con monitoring continuo delle performance.

3.2 Configurazione degli ottimizzatori

Il training utilizza Adam Optimizer con parametri beta personalizzati per migliorare la stabilità nella generazione.

Il learning rate di base è 0.0001 ed è gestito da un ReduceLROnPlateau Scheduler che riduce in maniera automatica il Learning Rate quando la Validation Loss smette di migliorare.

Per il training adversariale opzionale, il discriminatore utilizza un learning rate doppio rispetto al generatore, seguendo le pratice comuni per GAN.

Questa configurazione bilancia la competizione tra generatore e discriminatore.

4.3 Sistema di Loss Composite

La strategia di loss combina obiettivi multipli per catturare diversi aspetta della qualità delle immagini.

La loss primaria è L1(Mean Absolute Error) tra l'immagine generata e il target, scelta per ridurre la sfocatura rispetto alle loss L2.

Il sistema poi integra loss percettuale LPIPS opzionale per catturare similarità semantica oltre la similarità pixel-wise.

I pesi configurabili permettono di bilanciare reconstruction(15.0), percettuale(2.5) e adversariale(0.5) secondo le necessità del training.

4.4 Monitoring e Validazione

Il sistema implementa monitoring completo tramite TensorBoard, tracciand i loss compoments, il learning rate e le metriche di validazione.

La validazione viene eseguita ogni 5 epoche, con calcolo di metriche globali: SSIM, PSNE, L1/L2 distance.

Il sistema genera automaticamente campioni di esempio ogni 5 epoche per monitorare il processo visivamente.

Il salvataggio automatico del miglior modello basato su validation loss garantisce che vengano preservati i risultati ottimali durante la fase di training.

Sistema di Valutazione

5.1 Metriche Quantitative Implementate

Il sistema di valutazione calcola metriche multiple per fornire una valutazione globale della qualità di generazione.

Le metriche implementate includono:

- SSIM per similarità strutturale
- PSNR per qualità di ricostruzione
- Distanze L1 e L2 per errori pixel-wise
- LPIPS per qualità percettuale

Per valutazioni più avanzate, il sistema calcola FID, ossia il Fréchet Inception Distance, utilizzando InceptionV3 per confrontare distribuzioni di features tra immagini reali e generate.

L'Inception Score valuta simultaneamente qualità e diversità delle generazioni.

5.2 Implementazione Robusta delle Metriche

Ogni metrica è implementata con error handling completo e valori di fallback per garantire continuità della valutazione anche in caso di problemi computazionali.

Il sistema gestisce automaticamente conversioni di formato e normalizzazioni necessarie per il calcolo corretto delle metriche.

La classe FIDCalculator implementa il calcolo di FID robusto con fallback se InceptionV3 non dovesse essere disponibile.

In maniera simile, tutte le metriche hanno implementazioni alternative per gestire il funzionamento anche su sistemi con librerie limitate.

5.3 Valutazione Qualitativa e Interpretabilità

Oltre alle metriche numeriche, il sistema fornisce strumenti per la valutazione qualitativa attraverso le visualizzazioni.

Il sistema infatti genera griglie di confronto tra immagini reali e generate, permettendo la valutazione visiva diretta della qualità.

La funzionalità di visualizzazione attention permette di analizzare come il modello interpreta le descrizioni testuali, mostrando heatmap degli attention weights.

Questo fornisce una comprensione chiara sull'interpretibilità del modello e aiuta nel debugging dell'architettura.

Demo Interattiva Gradio

6.1 Interfaccia Utente

La Demo implementata tramite Gradio fornisce un'interfaccia web professionale per interagire con il modello addestrato.

L'applicazione include tre tab principali:

- Generazione Singola
- Generazione batch
- Visualizzazione Metriche

6.2 Funzionalità Avanzate

La modalità batch permette la generazione di variazioni multiple a partire da più descrizioni simultaneamente, dimostrando la capacità del modello di produrre output differenti.

Il sistema aggiunge in maniera automatica annotazioni alle immagini generate per identificarle.

La visualizzazione delle metriche carica automaticamente i risultati dell'ultima valutazione, presentando sia i grafici che le informazioni dettagliate sul modello.

Questo, quindi, fornisce trasparenza completa sulle performance e sulla configurazione del sistema.

6.3 Valore Dimostrativo

La demo rappresenta il successo completo dell'implementazione, dal momento che fornisce un'interfaccia accessibile per la sperimentazione del modello.

Mettendo da parte la qualità delle generazioni, la demo dimostra in maniera chiara il funzionamento di un sistema text-to-image completo.

L'interfaccia permette agli utenti di comprendere in maniera intuitiva le limitazioni del modello, osservando come diverse descrizioni influenzino l'output generato. Questo ha valore educativo significativo per comprendere i sistemi di Intelligenza Artificiale generativa.

Risultati e Analisi

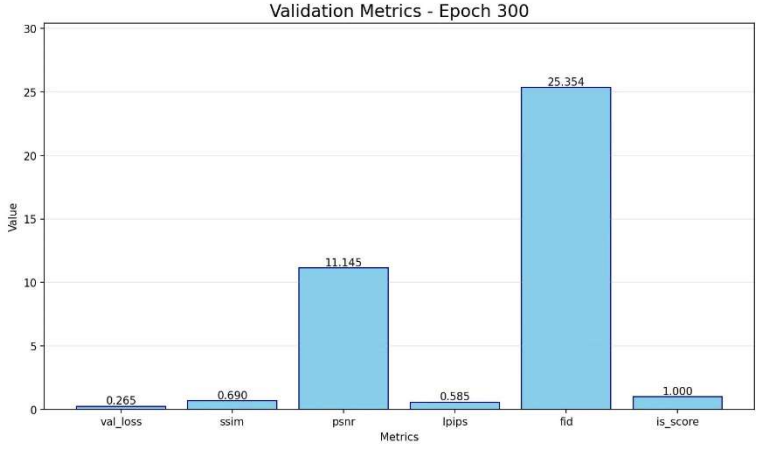
Il progetto ha completato con successo 300 epoche di training, raggiungendo una convergenza stabile e producendo output quantificabili.

L'analisi dei risultati combina la valutazione quantitativa tramite metriche standard e valutazione qualitativa attraverso confronto visivo diretto tra sprite reali e generati.

Il modello genera in maniera consistente immagini a partire da input testuali, dimostrando che l'architettura encoder-decoder implementata funziona correttamente e che l'apprendimento della mappatura testo-immagine avviene a livello operativo.

7.1 Performance Metriche Quantitative

I risultati numerici finali ci offrono una valutazione oggettiva delle performance:



Convergenza e stabilità:

- Validation Loss: 0.265 – indica una convergenza stabile senza overfitting dopo 300 epoche
- Training robusto completato senza failure catastrofici o instabilità

Qualità Strutturale:

- SSIM: 0.690 - similarità strutturale moderata-buona, indica che il modello riesce a catturare i layout e le forme base
- PSNR: 11.145 dB – qualità di ricostruzione pixel-wise limitata, conferma le differenze nei dettagli fini

Qualità Percettuale:

- LPIPS: 0.585 – Distanza percettuale moderata, suggerisce che le immagini sono visualmente interpretabili
- FID: 25.354 – Score elevato indica distribuzione statistica diversa dalle immagini reali
- Inception Score: 1.000 Score basso riflette limitazioni in qualità fine e diversità

7.2 Confronto Visivo: Immagini Reali vs Immagini Generate

La griglia di confronto fornisce un'evidenza visiva diretta, dandoci la possibilità di fornire una valutazione umana dei risultati.

Notiamo la notevole differenza tra le immagini reali e quelle generate; infatti, le immagini reali hanno una definizione nitida, palette vivaci e una riconoscibilità immediata, oltre ad una consistenza stilistica.

Gli Sprites generati invece hanno forme astratte, con colori presenti ma attenuati e sono figure “interpretabili”, ma posseggono tuttavia uno stile artistico astratto.

7.3 Correlazione tra le metriche

Effettuando una comparazione tra le due metriche, notiamo come il valore 0.69 di SSIM indica strutture di base riconoscibili, anche se non sempre si riesce identificare da uno sprite la struttura del Pokémon.

Il valore di PSNR 11.14 indica la perdita dei Dettagli Fini, che si manifesta visivamente nella sfocatura, la perdita di texture e nei contorni meno definiti.

Il valore LPIPS pari a 0.58 dovrebbe indicare che le immagini dovrebbero essere percettualmente interpretabili come creature.

Il valore FID pari a 25.35 indica il Gap stilistico evidente, che manifesta la differenza tra uno Sprite Professionale ed una sorta di “schizzo artistico”.

7.4 Successi Tecnici

Dal punto di vista tecnico e implementativo i traguardi raggiunti sono i seguenti:

Sistema end-to-end operativo con 38.8M parametri (27.7 M training) gestiti in maniera efficace.

Architettura stabile: BERT-mini(512dim) + CNN generatore con 768 canali base funziona senza errori

Training convergente: 300 epoche completate con monitoring continuo

Dal punto di vista dell’apprendimento invece possiamo notare che il modello riesce a correlare input testuali a output visivi distintivi, e la maggior parte delle volte descrizioni diverse producono immagini diverse.

Conclusioni e Sviluppi Futuri

8.1 Riepilogo Risultati

L'obiettivo principale, ossia quello di implementare un sistema completo text-to-image generation è stato raggiunto dal punto di vista implementativo.

Infatti, il modello riesce a generare immagini a partire da descrizioni testuali, dimostrando quindi che l'architettura encoder-decorder con il meccanismo di attention funziona correttamente.

I successi tecnici quindi sono:

- Implementazione completa di architettura encoder-decoder moderna (38.8M parametri)
- Training stabile per 300 epoche con convergenza a validation loss 0.265
- Sistema end-to-end funzionante dal preprocessing alla demo interattiva Gradio
- Apprendimento strutturale verificato da SSIM 0.69 e qualità percettuale LPIPS 0.58
- Pipeline robusta con gestione errori, monitoring e valutazione globale

8.2.1 Problemi e Sfide

Sfide Implementative:

Complessità architetturale:

Durante l'implementazione, la sincronizzazione tra componenti testuali e visuali ha presentato difficoltà notevoli.

L’allineamento tra output del text-encoder BERT e l’input del generatore CNN rispettivamente a dimensione variabile e fissa ha richiesto multiple iterazioni per trovare la configurazione corretta.

Instabilità del Training Generativo:

Il training di modelli generativi ha mostrano un’instabilità tipiga dei sistemi GAN.

La componente adversariale opzionale ha richiesto un bilanciamento attento dei learning rates e dei pesi delle loss funcitons.

Sono stati necessari diversi esperimenti per trovare una configurazione più o meno stabile.

Sfide Pre-processing:

Durante il preprocessing non sono stati individuate particolari difficoltà.

8.2.2 Limitazioni architettura attuale

Gap Qualitativo Fondamentale:

Il PSNR basso pari a 11.14 dB e il FID elevato pari a 25.35 indicano che l'architettura CNN tradizionale utilizzata non è ottimale per la generazione di sprite high-quality. L'approccio upsampling progressivo con blocchi residuali, pur tecnicamente corretto, non cattura la complessità stilistica richiesta.

Limitazioni del Meccanismo di Attention:

Sebbene il sistema di multi-head attention funzioni correttamente, l'analisi degli attention weights suggerisce che il modello non riesce a correlare specifiche parole chiave con regioni precise dell'immagine con la precisione necessaria per generazioni fedeli.

8.3 Miglioramenti futuri

Per migliorare il modello in un futuro bisognerebbe adottare l'utilizzo di architetture più moderne, come per esempio Diffusion Models o Vision Transformers, che rappresenterebbero un upgrade più significativo per migliorare la qualità delle generazioni.

Anche per ottimizzare il training si potrebbero adottare novità, come per esempio Loss functions avanzate come CLIP loss e style loss per migliorare l'allineamento semantico e la coerenza stilistica.

Inoltre, si potrebbe adottare il controllo condizionale, per specificare esplicitamente attributi come colore, tipo e dimensione.

8.4 Conclusione

Il progetto PikaPikaGenerator ha raggiunto con successo l'obiettivo principale di implementare un sistema completo di text-to-image generation funzionante. Sebbene gli sprite generati differiscano significativamente dagli originali in termini di qualità visiva, il sistema dimostra apprendimento effettivo della mappatura testo-immagine e produce output visivamente interpretabili.

Bibliografia

[1] The Complete Pokédex Dataset https://github.com/cristobalmitchell/pokedex/