Generazione di cartelli stradali tramite cGAN e valutazione con rete neurale

Maria Francesca Merangolo
Politecnico di Torino
s318905@studenti.polito.it

Stefano Dellarossa
Politecnico di Torino
s305940@studenti.polito.it

Luca Delfino
Politecnico di Torino
s295970@studenti.polito.it

Abstract

Negli ultimi anni le reti neurali generative avversarie condizionali (cGAN) hanno rivoluzionato l'intelligenza artificiale, offrendo strumenti avanzati per la generazione di immagini sintetiche di alta qualità. Questo lavoro si focalizza sull'applicazione delle cGAN per la generazione di cartelli stradali russi, con l'obiettivo di creare un ampio dataset sintetico utile per l'addestramento di modelli di visione artificiale, come quelli impiegati nei veicoli autonomi. Il modello è stato addestrato per produrre segnali appartenenti a diverse categorie con un approccio ottimizzato grazie alla sperimentazione su diversi iperparametri, tra cui dimensione del vettore latente, batch size e learning rate.

Le performance sono state valutate tramite metriche quali F1-score, Fréchet Inception Distance (FID) e la funzione di loss, confrontando le immagini generate con quelle reali. Inoltre, una rete neurale ResNet18 è stata implementata per classificare le immagini sintetiche, fornendo ulteriori insight sulla qualità del modello e sulle sue potenzialità. I risultati dimostrano l'efficacia delle cGAN nella generazione di segnali stradali, evidenziando sfide legate alla generalizzazione del modello e suggerendo direzioni future per miglioramenti.

1. Introduzione

Il progetto prevede l'addestramento di una cGAN su un dataset reale di segnali stradali russi, suddivisi in 8 categorie funzionali: "Pericolo", "Priorità", "Divieto", "Obbligo", "Regolamentazione", "Informazione", "Servizi", "Pannelli Integrativi". Utilizzando le etichette di classe come input condizionali, il generatore della cGAN è in grado di creare immagini coerenti con le caratteristiche visive tipiche di ciascuna categoria. Oltre a generare le immagini, il sistema viene valutato attraverso l'implementazione di una rete neurale convoluzionale (ResNet18), che classifica le immagini generate, contribuendo così a misurare la coerenza e l'accuratezza del modello.

La scelta di utilizzare una cGAN è particolarmente utile quando si cerca di generare dati realistici che possano essere utilizzati per addestrare modelli di visione artificiale in scenari complessi come in questo contesto di classificazione e riconoscimento di segnali stradali.

2. Descrizione del dataset

2.1. Origine dei dati

Il dataset utilizzato per questo progetto, disponibile pubblicamente, è stato ottenuto dalla piattaforma Kaggle [1]. Ogni sottocartella del dataset corrisponde ad una specifica categoria di cartello. Per questo progetto, è stato utilizzato l'intero dataset disponibile, che offre un'ampia varietà di cartelli rappresentativi, ideale per addestrare una cGAN.

2.2. Ricerca degli Iperparametri

Per migliorare le prestazioni della nostra cGAN, abbiamo condotto un'ampia ricerca degli iperparametri. Gli iperparametri considerati sono stati:

- **Noise vector (dimensione del vettore latente):** Abbiamo sperimentato diverse dimensioni per il noise vector, variando tra 100 e 256.
- **Batch size:** Sono stati testati valori di batch size di 64, 128 e 256 per ottimizzare il trade-off tra velocità di addestramento e stabilità del modello.
- **Learning rate:** Il learning rate è stato variato tra 0.0001 e 0.0002 per identificare il valore che fornisse la miglior convergenza.
- **Numero di feature maps:** Abbiamo testato diverse configurazioni per il numero di feature maps nei layer convoluzionali del generatore e del discriminatore, variando il numero di filtri per bilanciare complessità e accuratezza del modello (n1 = 512 e 1024).

2.3. Caratteristiche del dataset

Il dataset comprende immagini di 8 diversi cartelli stradali russi, ciascuno rappresentato da 10.000 immag-

ini, per un totale di 80.000 immagini. Tutte le immagini sono in formato RGB (a tre canali) e sono state ridimensionate a 64x64 pixel per garantire uniformità durante l'addestramento e ridurre la complessità computazionale. Questo formato è stato scelto per mantenere una risoluzione sufficiente per una classificazione accurata.

Inoltre il dataset risulta diviso per il 70% delle immagini per il training e il 30% per la validation.

Di seguito sono riportate alcune delle caratteristiche principali del dataset:

- **Numero di classi (cartelli autostradali russi):** 8
- **Numero totale di immagini:** 80.000
- **Immagini per classe:** 10.000
- **Dimensione immagine:** 64x64 pixel
- **Formato colore:** RGB
- **Batch size:** 128 (utilizzato durante l'addestramento)
- **Dimensione del vettore latente (nz):** 100
- **Learning rate:** 0.0002

2.4. Pulizia e Preprocessing dei dati

Prima dell'addestramento, il dataset è stato preprocessato come segue:

- **Formato RGB:** Tutte le immagini sono state convertite in formato RGB.
- **Ridimensionamento:** Le immagini sono state uniformate a 64×64 pixel.
- **Normalizzazione:** Le immagini sono state normalizzate con media e deviazione standard di $(0.5, 0.5, 0.5)$ per ciascun canale RGB.

2.5. Data Augmentation

Dato che il dataset era già ampio, l'uso della data augmentation è stato limitato. Sono state applicate solo le seguenti trasformazioni:

- **Ridimensionamento:** Uniformato a 64×64 pixel.
- **Normalizzazione:** Media e deviazione standard di $(0.5, 0.5, 0.5)$ per ogni canale RGB.
- **Tecniche commentate:** Trasformazioni come l'affinamento casuale e il flip orizzontale sono state commentate nel codice e potrebbero essere esplorate in futuri esperimenti.

La decisione di limitare la data augmentation è stata presa per evitare variazioni eccessive, date le dimensioni adeguate del dataset.

3. Metodo - parte 1: cGAN

3.1. Approccio e architettura della Rete

L'approccio impiegato per generare immagini condizionate prevede l'addestramento di un Generatore e di un Discriminatore che competono tra loro, migliorando nel tempo la qualità delle immagini generate. Il Generatore crea immagini a partire da rumore casuale concatenato alle etichette di classe, mentre il Discriminatore valuta la coerenza di queste immagini. Di seguito, descriviamo in dettaglio le architetture delle reti e la metodologia di addestramento.

3.1.1 Generatore

Il Generatore è progettato per prendere come input un vettore di rumore e un'etichetta di classe. È composto da una serie di strati di convoluzione trasposta che riscalano l'input di rumore alla dimensione dell'immagine desiderata. L'architettura è la seguente:

- Input: Vettore di rumore e embedding dell'etichetta di classe
- Strati: Diversi strati di convoluzione trasposta con attivazioni ReLU e Normalizzazione per Batch
- Output: Immagine generata di dimensioni $64 \times 64 \times 3$

3.1.2 Discriminatore

Il Discriminatore prende un'immagine e la relativa etichetta di classe come input e classifica l'immagine come reale o falsa. Questa include:

- Input: Immagine e embedding dell'etichetta di classe
- Strati: Strati convoluzionali con attivazioni LeakyReLU e un'uscita finale sigmoidale
- Output: Punteggio di probabilità. Questa indica la probabilità che l'immagine sia reale

3.2. Modello e Funzioni di Loss

La funzione di loss utilizzata per la rete cGAN è la Binary Cross Entropy (BCE), che serve a distinguere tra immagini reali e generate. La loss del Discriminatore è la somma delle loss calcolate sulle immagini reali e su quelle generate, dove il Discriminatore cerca di assegnare rispettivamente le etichette corrette di 'reale' o 'falso'. La loss del Generatore, invece, è progettata per ingannare il Discriminatore, cercando di far classificare le immagini generate come reali.

3.3. Selezione degli Iperparametri

Gli iperparametri come i learning rate, le dimensioni dei batch e le dimensioni della rete sono stati selezionati in base ai risultati empirici e alle pratiche comuni nell'addestramento delle cGAN. Ad esempio:

- Learning rate: $\frac{0.0002}{2}$ per il Generatore e $0.0002 \cdot 2$ per il Discriminatore
- Dimensione del batch: 128

Queste scelte sono state affinati attraverso prove ed errori sul set di training.

3.4. Sfide e Soluzioni

Una sfida significativa è stata bilanciare l'addestramento del Generatore e del Discriminatore, che spesso portava a instabilità nel processo di training. Per mitigare ciò si possono utilizzare tecniche come il label smoothing e Learning Rate Scheduling.

3.5. Valutazione delle Prestazioni

Le prestazioni sono state valutate utilizzando diverse metriche:

- **Inception Score (IS)**: Misura la qualità delle immagini generate valutando il loro realismo e diversità.
- **Frechet Inception Distance (FID)**: Confronta la distribuzione delle immagini generate con quella delle immagini reali.

4. Metodologia - parte 2: ResNet18

4.1. Architettura della Rete

L'architettura del modello si basa su una rete neurale convoluzionale profonda (CNN), che ha dimostrato di essere efficace nei compiti di classificazione delle immagini. La rete è composta da diversi strati convoluzionali, seguiti da strati di max-pooling e da strati completamente connessi. L'ultimo strato è un classificatore softmax che restituisce la distribuzione di probabilità tra le classi. Dopo ogni strato convoluzionale abbiamo impiegato la normalizzazione dei batch per stabilizzare e accelerare il processo di addestramento.

Abbiamo optato per una CNN poiché offre ottime prestazioni nei task legati alle immagini grazie alla sua capacità di apprendere gerarchie spaziali di caratteristiche. Approcci alternativi, come le reti completamente connesse, sono stati scartati poiché generalmente mostrano prestazioni scarse su dati di input ad alta dimensionalità, come le immagini.

L'implementazione è stata basata su un framework [3], che abbiamo esteso introducendo livelli personalizzati e modificando l'architettura per adattarla meglio al nostro problema.

4.2. Funzione di Loss e Iperparametri

Abbiamo utilizzato la *cross-entropy loss* come funzione di loss principale. La selezione degli iperparametri è stata effettuata utilizzando una combinazione di ricerca a griglia e ottimizzazione manuale. Abbiamo utilizzato un learning rate pari a 0.001 e batch di 128.

4.3. Valutazione delle Prestazioni

Le prestazioni del modello sono state valutate utilizzando diverse metriche, tra cui accuracy, precision, recall, F1-Score e loss per valutare la qualità dei campioni generati.

5. Esperimenti - parte 1: cGAN

In questa sezione riportiamo i risultati degli esperimenti condotti per valutare le prestazioni del nostro modello. Le metriche utilizzate per l'analisi sono l'Inception score, il Fréchet Inception Distance (FID) e la funzione di loss.

5.1. Configurazione Ottimale

La configurazione con le migliori prestazioni è stata ottenuta utilizzando un batch size di 128, noise vector di 100 e un learning rate di $2 * 10^{-4}$. La tabella 1 mostra i risultati ottenuti per questa configurazione.

Iperparametri	Inception score	FID	Loss
Ottimale	1.16	12.11	G: 6.86, D: 0.19

Table 1. Risultati della configurazione ottimale

La Figura 1 mostra un esempio di immagini di cartelli stradali generate dalla cGAN, addestrata con le configurazioni ottimali.

Figure 1. Esempi di risultati ottenuti dal modello con etichette

La Figura 2 mostra la variazione della FID nel tempo. Un andamento discendente della FID indica un miglioramento nella qualità delle immagini generate.

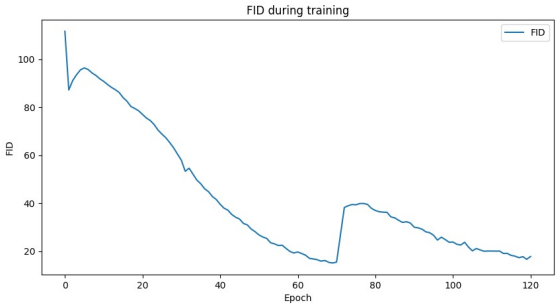

Figure 2. Curva della FID per il set di addestramento

5.2. Curva di Loss

Le curve di loss per il generatore e discriminatore sono mostrate nella Figura 3. Nelle prime epoche, la loss del generatore inizia da valori relativamente bassi e tende ad aumentare progressivamente. Questo suggerisce che inizialmente il generatore produce immagini che il discriminatore riesce a identificare facilmente come false, un comportamento tipico delle prime fasi di addestramento. D'altra parte, la loss del discriminatore parte da valori leggermente più alti rispetto a quella del generatore, ma diminuisce rapidamente. Questo indica che il discriminatore diventa rapidamente abile nel distinguere le immagini reali da quelle generate, mostrando un miglioramento continuo nelle ultime iterazioni e stabilizzandosi su valori molto bassi.

Per il generatore, tuttavia, si osservano oscillazioni più ampie, ma stabili, dopo circa 30.000 iterazioni. Questo comportamento riflette un equilibrio instabile tra i due modelli: sebbene il generatore migliori nella qualità delle immagini prodotte, le oscillazioni suggeriscono che il sistema è in uno stato di equilibrio dinamico in cui il generatore e il discriminatore si sfidano costantemente, influenzando le performance reciproche.

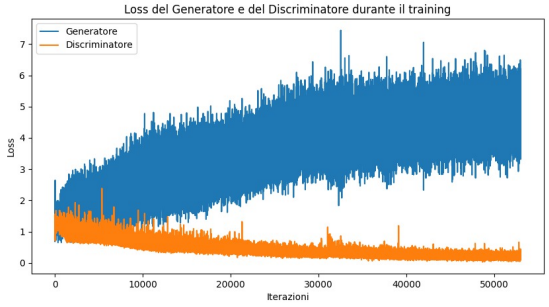

Figure 3. Curva di loss del generatore e del discriminatore

5.3. Studio di Ablazione

Per valutare l'impatto di singole componenti del sistema, abbiamo effettuato uno studio di ablazione rimuovendo la data augmentation in quanto avevamo molti dati nel nostro dataset garantendoci una qualità maggiore a parità di epoche. I risultati ottenuti dimostrano che in situazioni in cui il dataset è già ricco di dati il data augmentation potrebbe far esplodere la complessità della rete non garantendo risultati migliori.

5.4. Confronto con Diversi Iperparametri e Architetture

Abbiamo esplorato diverse scelte di iperparametri e architetture. Le modifiche nelle due configurazioni di prova sono state:

- Non ottimale 1:
  - L'architettura del generatore tale che il layer più esterno avesse $n = 1024$, con gli altri iperparametri uguali alla configurazione ottimale.
- Non ottimale 2:
  - Noise vector: 100
  - Batch size: 256
  - Learning rate: $2 * 10^{-5}$
- Non ottimale 3:
  - Noise vector: 100
  - Batch size: 128
  - Learning rate: $1 * 10^{-5}$
- Non ottimale 4:
  - Noise vector: 256
  - Batch size: 128
  - Learning rate: $2 * 10^{-5}$
- Non ottimale 5:
  - Noise vector: 100
  - Batch size: 64
  - Learning rate: $2 * 10^{-5}$

La Tabella 2 riassume i risultati ottenuti per le diverse configurazioni.

Iperparametri	Inception score	FID	Loss
Non ottimale 1	3.54	56.82	G: 14.78, D: 0.01
Non ottimale 2	1.12	18.43	G: 57.72, D: 0.01
Non ottimale 3	1.17	13.87	G: 17.02, D: 0.01
Non ottimale 4	1.13	9.93	G: 12.49, D: 0.03
Non ottimale 5	1.12	9.20	G: 10.81, D: 0.01

Table 2. Risultati per diverse scelte di iperparametri e architettura dopo 140 epoche

5.5. Visualizzazione e Discussione dei Fallimenti

Abbiamo utilizzato tecniche di visualizzazione per comprendere meglio il funzionamento del modello. Le Figure 4 e 5 mostrano alcuni esempi di successi e fallimenti del modello. In particolare, i casi di fallimento sono stati analizzati per comprendere le possibili cause, che includono una eccessiva complessità della rete che non genera in automatico dei risultati migliori, anzi abbiamo ottenuto i risultati migliori quando abbiamo tenuto bassa la complessità della rete.

Figure 4. Esempi di risultati ottenuti dal modello

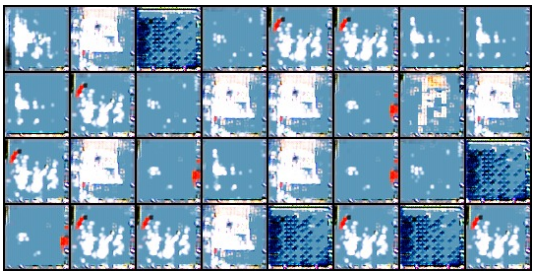

Figure 5. Esempi di fallimenti del modello

6. Esperimenti - Parte 2: ResNet18

In questa sezione vengono riportati i risultati degli esperimenti effettuati per valutare le prestazioni del modello. Il modello è stato addestrato per 10 epoche e testato sullo stesso dataset usato per la cGAN, e le metriche utilizzate per la valutazione sono precision, recall, accuracy, f1-score, confusion matrix e la funzione di Loss.

6.1. Configurazione Migliore

La configurazione con le migliori prestazioni è stata ottenuta utilizzando i seguenti iperparametri:

- Batch size: 128
- Learning rate: 0.001

. In Tabella 3, riportiamo le metriche ottenute dalla configurazione ottimale. Inoltre, la Figura 7 mostra le curve di Loss per il set di training e quello di validation. Il validation è costituito da 1024 immagini fake, generate precedentemente dalla rete cGan.

Metrica	Real Validation Set	Fake Validation Set
Precision mean	0.9959	1.0000
Recall mean	0.9958	1.0000
Accuracy mean	0.9958	1.0000
F1-Score mean	0.9958	1.0000
Loss	0.0125	0.0057

Table 3. Prestazioni della configurazione migliore dopo 10 epoche.

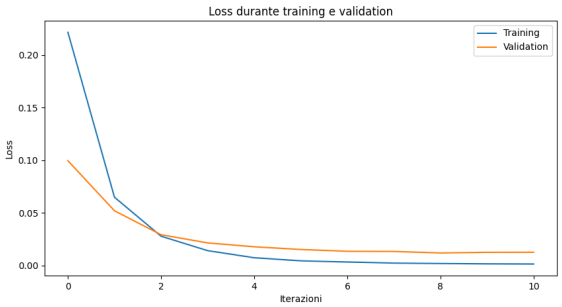

Figure 6. Curve di Loss per il set di training e validation reale.

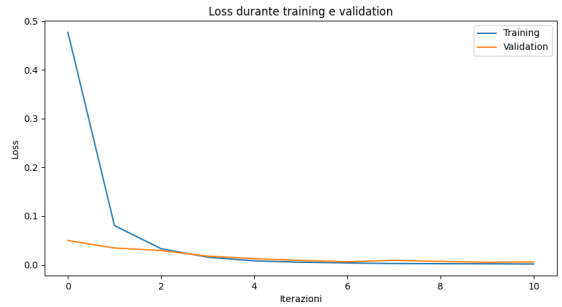

Figure 7. Curve di Loss per il set di training e validation sintetico.

6.2. Confronto tra immagini reali e immagini sintetiche

Abbiamo valutato la coerenza delle immagini sintetiche analizzando le prestazioni attraverso le seguenti metriche:

- Precision
- Recall
- Accuracy
- F1-Score

6.3. Analisi delle metriche e delle criticità

Abbiamo analizzato il comportamento delle metriche per ogni classe separatamente, evidenziando come alcuni tipi di segnali risultino più difficili da riconoscere rispetto ad altri.

Abbiamo testato la rete prima attraverso un set di immagini reali e poi con un set di immagini sintetiche generate dalla nostra cGan.

6.3.1 Precision

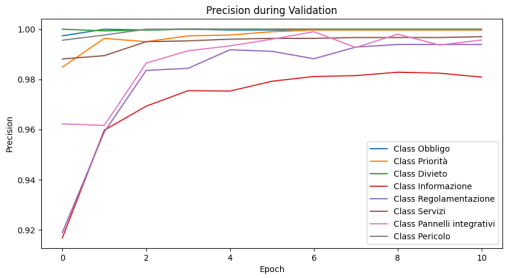

Figure 8. Precision per ogni classe durante il validation (reale).

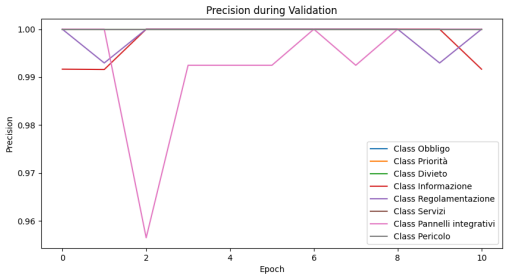

Figure 9. Precision per ogni classe durante il validation (sintetico).

6.3.2 Recall

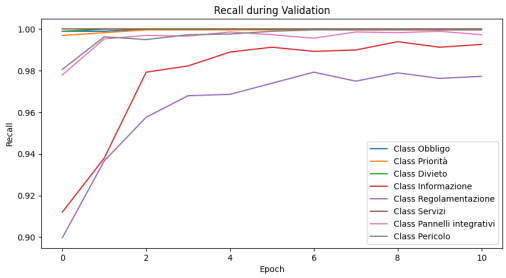

Figure 10. Recall per ogni classe durante il validation (reale).

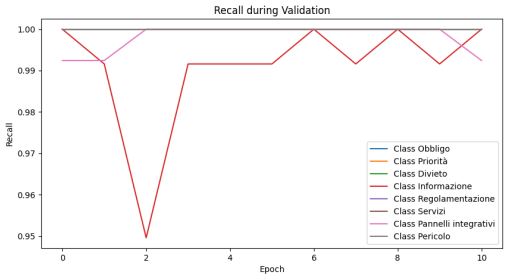

Figure 11. Recall per ogni classe durante il validation (sintetico).

6.3.3 F1-Score

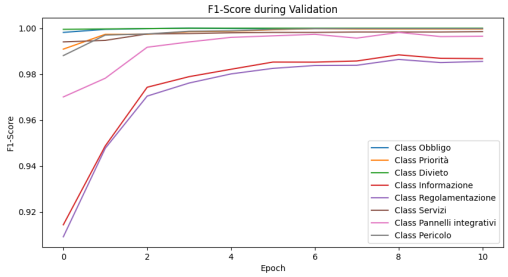

Figure 12. F1-score per ogni classe durante il validation (reale).

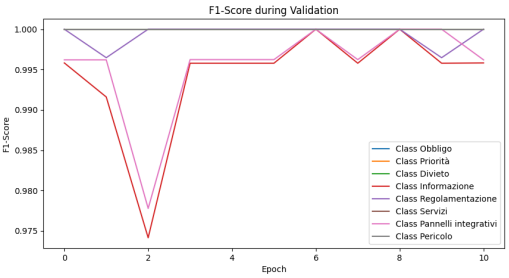

Figure 13. F1-Score per ogni classe durante il validation (sintetico).

6.3.4 Accuracy

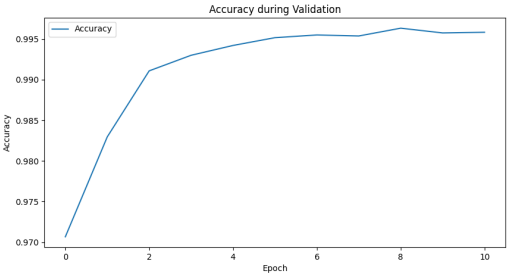

Figure 14. Media dell'accuracy durante il validation (reale).

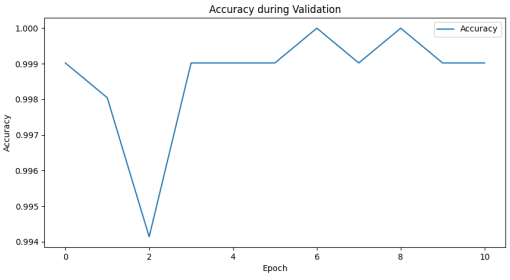

Figure 15. Media dell'accuracy durante il validation (sintetico).

In generale, le 4 metriche dimostrano come il modello riesca a discriminare in maniera soddisfacente le 8 tipologie di segnali dopo un addestramento di 10 epoche. Infatti, sia

nel caso del validation reale sia nel caso di quello sintetico, i diversi tipi di segnali vengono classificati correttamente, non evidenziando quasi alcuna differenza tra i due e rendendoci soddisfatti della coerenza delle immagini sintetiche. Più in particolare, i grafici mettono a risalto una lieve criticità nel saper classificare nel corretto modo i segnali di Informazione, Regolamentazione e Pannelli integrativi nelle prime fasi dell'addestramento. Non a caso, anche graficamente, sono i segnali più difficili e complessi da generare e classificare, e ci aspettavamo un risultato simile.

6.3.5 Confusion matrix

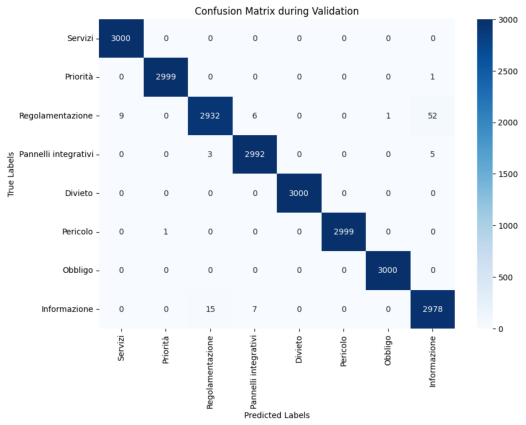

Figure 16. Confusion matrix per ogni classe alla epoca 10 (validation reale).

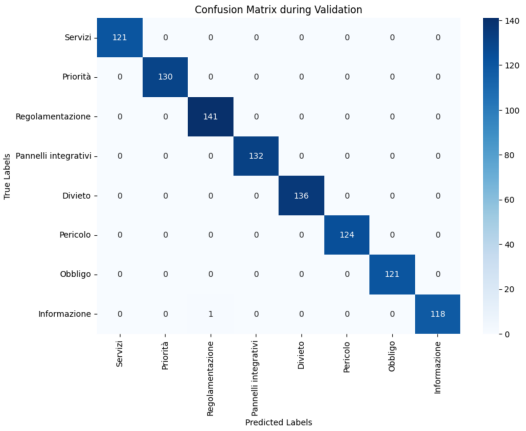

Figure 17. Confusion matrix per ogni classe alla epoca 10 (validation sintetico).

Le matrici di confusione confermano quanto descritto sopra. In particolare, nel caso del validation di immagini sintetiche, la percentuale di successo è quasi del 100%, con un solo caso di segnale classificato male.

7. Conclusioni

In questo progetto, abbiamo ottenuto risultati significativi in termini di FID, per la prima parte, e F1-Score, precision, recall, accuracy e loss, per la seconda parte, riuscendo a identificare la migliore configurazione del modello.

Una delle principali sfide affrontate è stata la selezione dei giusti iperparametri e la gestione della complessità del modello. Abbiamo risolto queste difficoltà con un approccio sistematico di tuning e studio di ablazione, che ci ha permesso di comprendere meglio quali componenti del modello fossero cruciali per il suo funzionamento.

Tuttavia, alcuni aspetti richiedono ulteriori miglioramenti. Ad esempio, in certi casi il modello non ha generalizzato bene su dati non visti, suggerendo che potrebbe essere necessaria una migliore regolarizzazione o tecniche per mantenere più stabili e piccole le loss.

Per il futuro, sarebbe interessante esplorare nuove tecniche e strategie per tenere sotto controllo le loss e di conseguenza aumentare la qualità delle immagini prodotte, seppur soddisfacenti. Inoltre, l'integrazione di una più approfondita analisi dei fallimenti del modello potrebbe fornire spunti utili per migliorare la robustezza del sistema.

References

[1] Sergey Kulakin. Russian Road Signs Categories Dataset. Disponibile su: https://www.kaggle.com/datasets/sergeykulakin/russian-road-signs-categories-dataset. 1

[2] PyTorch. DCGAN Tutorial: Training a Generative Adversarial Network (GAN) on Faces Dataset. Disponibile su: https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html.

[3] PyTorch. ResNet: Deep Residual Learning for Image Recognition. Disponibile su: https://pytorch.org/hub/pytorch_vision_resnet/. 3