Recunoașterea Expresiilor Faciale utilizând Vision Transformers și Transfer Learning

Miloiu Cristi-Constantin, Popa Cătălina-Florentina, Popa Irina-Ștefania
Facultatea de Electronică, Telecomunicații și Tehnologia Informației
Universitatea Națională de Știință și Tehnologie POLITEHNICA București
București, România

***Abstract*—Acest proiect abordează problema recunoașterii automate a expresiilor faciale (Facial Emotion Recognition - FER) utilizând un set de date format din 3000 de imagini și arhitectura Vision Transformers (ViT). Soluția propusă se bazează pe paradigma "transfer learning", adaptând un model pre-antrenat pe setul ImageNet pentru a clasifica 7 tipuri de emoții specifice. Lucrarea detaliază etapele de preprocesare, strategiile de augmentare a datelor și ajustările arhitecturale implementate pentru a maximiza performanța în condițiile unui set de date limitat. De asemenea, este analizat impactul tehnicilor de testare avansate (TTA) asupra preciziei finale a sistemului.**

***Index Terms*—Vision Transformer, Transfer Learning, Recunoașterea Expresiilor Faciale, Augmentarea Datelor, Deep Learning**

I. Introducere

Proiectul își propune să rezolve o problemă interesantă din domeniul inteligenței artificiale: recunoașterea emoțiilor umane din imagini. Această tehnologie are multe utilizări practice, de la îmbunătățirea interacțiunii dintre oameni și computere, până la sisteme care pot detecta dacă un șofer este obosit sau distras. Setul de date pe care am lucrat conține 3000 de imagini alb-negru, destul de mici (48×48 pixeli), care reprezintă una dintre cele 7 emoții de bază: furie, dezgust, frică, fericire, neutru, tristețe și surpriză. Majoritatea imaginilor (2700) sunt folosite pentru a "învăța" modelul, iar restul (300) pentru a-l testa.

Clasificarea corectă a emoțiilor dintr-o imagine statică reprezintă o provocare majoră, dată fiind subtilitatea diferențelor dintre clase (de exemplu, frica versus surpriza) și variabilitatea trăsăturilor faciale. În plus, dimensiunea redusă a setului de antrenament predispune modelele profunde la fenomenul de overfitting, limitând capacitatea de generalizare.

Pentru a adresa aceste limitări, am optat pentru utilizarea unei arhitecturi pre-antrenate de tip Vision Transformer (ViT). Aceasta beneficiază de capacitatea de a extrage caracteristici vizuale complexe învățate pe seturi de date masive, fiind adaptată ulterior specificului recunoașterii faciale prin fine-tuning.

Pentru a optimiza procesul de învățare și a maximiza performanța, am utilizat diverse tehnici: am aplicat transformări asupra imaginilor de antrenament (rotații, oglindiri), am integrat metode de regularizare pentru a preveni memorarea mecanică și am asigurat o pondere egală a claselor. Obiectivul este dezvoltarea unui sistem robust, capabil să generalizeze corect pe date noi.

II. Preprocesarea Datelor

Înainte de a antrena modelul, datele trebuie pregătite ("curățate" și aduse la un format standard). Acesta este un pas esențial pentru ca algoritmul să funcționeze corect.

A. Transformarea imaginilor

Modelul pe care l-am ales (ViT) are nevoie de imagini care arată într-un anumit fel.

1) **Culoarea (Grayscale $\rightarrow$ RGB)**: Imaginile din setul de date sunt monocrome (un singur canal), însă arhitectura ViT pre-antrenată necesită imagini color (3 canale). Prin urmare, am replicat canalul unic de trei ori pentru a simula o imagine color, asigurând astfel compatibilitatea cu modelul fără a altera informația vizuală.
2) **Mărirea imaginilor**: Imaginile originale sunt foarte mici (48×48 pixeli). Modelul așteaptă imagini mult mai mari (224×224 pixeli). Am mărit imaginile folosind un algoritm de calitate (interpolare bicubică) care încearcă să nu piardă detaliile importante ale feței atunci când face poza mai mare.
3) **Aducerea la aceeași scară (Normalizare)**: Am transformat valorile pixelilor astfel încât să fie similare cu cele pe care le-a văzut modelul când a fost antrenat inițial. Asta ajută calculele matematice să meargă mai repede și mai stabil.

```
transforms.Compose([
    # Conversie la 3 canale (RGB)
    transforms.Grayscale(num_output_channels=3),
    # Redimensionare cu interpolare bicubica
    transforms.Resize((224, 224),
        interpolation=transforms.InterpolationMode.
    BICUBIC),
    transforms.ToTensor(),
    # Normalizare cu mediile ImageNet
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    )
])
```

Listing 1. Transformările Standard (fără Augmentare)

B. Creșterea artificială a setului de date (Augmentare)

Având un set de date limitat (2700 imagini), există riscul de *overfitting* (memorarea mecanică a exemplelor). Pentru a preveni acest fenomen, aplicăm transformări dinamice asupra imaginilor în timpul antrenamentului:

- **Oglindire**: Uneori îi arătăm poza inversată stânga-dreapta. Emoția rămâne aceeași, dar modelul vede o imagine ”nouă”.
- **Rotire**: Rotim ușor poza (cu maxim 15 grade) pentru ca modelul să recunoască o față chiar dacă persoana stă puțin înclinată.
- **Deplasare și Scalare**: Mutăm puțin fața în cadru sau facem zoom, ca modelul să nu se obișnuiască să găsească ochii sau gura mereu în același loc fix.
- **Luminozitate**: Schimbăm puțin cât de luminoasă este poza, pentru a simula diferite condiții de lumină.
- **Ștergere Aleatoare**: Uneori acoperim o mică parte din poză cu un pătrat gri. Asta forțează modelul să se uite la toată fața, nu doar la un singur detaliu.

```
transforms.Compose([
    # Simulare imagine color prin replicare
    transforms.Grayscale(num_output_channels=3),
    # Redimensionare pentru ViT (224x224)
    transforms.Resize((224, 224)),
    # Transformari geometrice
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.RandomRotation(15),
    transforms.RandomAffine(
        degrees=0,
        translate=(0.15, 0.15),
        scale=(0.9, 1.1)
    ),
    # Variatii de culoare
    transforms.ColorJitter(brightness=0.3, contrast
    =0.3),
    # Regularizare prin stergere
    transforms.RandomErasing(p=0.2)
])
```

Listing 2. Pipeline-ul de Augmentare (PyTorch)

C. Strategii Avansate de Regularizare

Am integrat tehnica **Mixup** [3], care generează exemple sintetice prin combinarea liniară a imaginilor și etichetelor ($\tilde{x} = \lambda x_i + (1 - \lambda)x_j$), coeficientul λ fiind extras dintr-o distribuție Beta(0.2, 0.2). Aceasta previne overfitting-ul pe clasele minoritare și îmbunătățește robustețea la granițele de decizie.

III. Metodologia și Arhitectura Modelului

Am folosit o tehnologie modernă care a schimbat recent felul în care computerele ”văd” imaginile, numită Transformer [2], [4].

A. Arhitectura Vision Transformer (ViT)

Spre deosebire de rețelele convoluționale (CNN) care procesează imaginea local, Vision Transformer-ul tratează imaginea ca o secvență de patch-uri, similar modului în care Transformerele procesează cuvintele în NLP.

1) **Patch Embedding**: Imaginea de input este divizată în patch-uri de dimensiune fixă (16×16 pixeli).
2) **Proiecția Liniară**: Fiecare patch este aplatizat și proiectat într-un spațiu vectorial latent.
3) **Positional Embedding**: Se adaugă informație despre poziția spațială a fiecărui patch, crucială pentru menținerea structurii imaginii.
4) **Mecanismul de Self-Attention**: Componenta centrală a modelului permite fiecărui patch să interacționeze cu toate celelalte.

```
# Incarcarea modelului pre-antrenat (Google ViT)
model = AutoModelForImageClassification.
    from_pretrained(
    "google/vit-base-patch16-224",
    num_labels=7,
    ignore_mismatched_sizes=True
)

# Inlocuirea clasificatorului original
# Adaugam Dropout pentru regularizare
model.classifier = nn.Sequential(
    nn.Dropout(p=0.3),
    nn.Linear(model.classifier.in_features, 7)
)
```

Listing 3. Inițializarea Modelului și Adaptarea Capului de Clasificare

Capul de clasificare original a fost înlocuit cu unul specific sarcinii noastre, adaptat pentru cele 7 clase de emoții.

B. Strategia de Antrenament

Procesul de antrenare a fost optimizat pentru a combate dezechilibrul claselor și a asigura stabilitatea convergenței.

1) **Weighted Random Sampling**: Pentru a contracara dezechilibrul distribuției claselor (ex. predominanța clasei ”Happy”), am implementat o schemă de eșantionare ponderată, asigurând că fiecare batch conține o reprezentare echilibrată a tuturor emoțiilor.
2) **Discriminative Learning Rates**: Am aplicat rate de învățare diferențiate: mai mici pentru straturile backbone-ului (pentru a prezerva trăsăturile generale) și mai mari pentru noul clasificator (pentru o adaptare rapidă la sarcina curentă).
3) **Label Smoothing**: Pentru a preveni supra-încrederea modelului în predicții (over-confidence), am utilizat etichete ”soft” ($\epsilon = 0.1$). Această tehnică penalizează certitudinea extremă și îmbunătățește calibrarea modelului pe date ambigue.

C. Detalii de Implementare

Sistemul a fost implementat utilizând framework-ul PyTorch. Aspectele cheie ale configurației includ:

- **Optimizator**: Am utilizat algoritmul **AdamW** [5], care combină avantajele metodei Adam cu o decuplare corectă a regularizării Weight Decay (0.01), asigurând o generalizare superioară.
- **Eșantionare Ponderată (Math)**: Pentru a balansa clasele, greutatea fiecărei etichete a fost calculată utilizând formula amortizată: $w_c = \sqrt{\frac{N_{total}}{N_{clasa}}}$. Aceasta previne supra-reprezentarea claselor minoritare care ar putea duce la overfitting pe acestea.
- **Scheduler**: Rata de învățare este ajustată dinamic folosind *ReduceLROnPlateau*. Dacă eroarea pe setul de validare nu scade timp de 3 epoci consecutive (patience=3), rata de învățare este înjumătățită (factor=0.5), permițând o ajustare fină a greutăților în fazele finale ale antrenamentului.

D. Mediu Experimental

Antrenamentul a fost realizat utilizând *Metal Performance Shaders (MPS)* pentru accelerare GPU și *Mixed Precision Training (FP16)* pentru eficiență. Hyperparametrii cheie: Batch Size = 32, 30 Epoci, Learning Rate = 2×10^{-5}.

E. Optimizarea în Etapa de Testare (TTA)

Pentru a îmbunătăți robustețea predicțiilor finale, am adoptat o strategie de Test-Time Augmentation. Procesul implică:

- Evaluarea imaginii originale.
- Evaluarea versiunii oglindite (flip orizontal).
- Evaluarea unei versiuni ușor scalate (zoom).

Rezultatul final se obține prin medierea probabilităților oferite de model pentru aceste variații.

```
probs_sum = None

# Iteram prin transformarile TTA (Original, Flip, Crop)
for tta_transform in tta_transforms:
    # Aplicam transformarea si obtinem predictia
    img_tensor = tta_transform(image)
    outputs = model(img_tensor).logits

    # Calculam probabilitatile (Softmax)
    probs = torch.softmax(outputs, dim=1)

    # Acumulam rezultatele
    if probs_sum is None:
        probs_sum = probs
    else:
        probs_sum += probs

# Facem media probabilitatilor pentru decizia finala
avg_probs = probs_sum / len(tta_transforms)
```

Listing 4. Implementarea TTA (Pseudo-cod simplificat)

Această metodă funcționează similar unui ansamblu de modele, reducând erorile izolate și crescând încrederea în clasificarea corectă.

IV. Rezultate și Concluzii

A. Analiza Performanței

Antrenamentul modelului s-a desfășurat pe parcursul a 30 de epoci. Monitorizarea metricilor de performanță a indicat o convergență stabilă, tehnicile de regularizare prevenind eficient fenomenul de overfitting. Deși acuratețea pe setul de antrenament a crescut rapid, diferența față de setul de validare s-a menținut în limite acceptabile, demonstrând capacitatea de generalizare a rețelei.

Rezultatul final obținut pe setul de testare este o acuratețe de **62.67%**. Pentru a evalua corect această performanță, trebuie să considerăm următorii factori de referință:

- **Probabilitatea Aleatoare (Baseline)**: Într-o problemă de clasificare cu 7 clase, alegerea aleatoare ar rezulta într-o acuratețe de aproximativ 14.28%. Performanța modelului este semnificativ superioară acestui prag.
- **Acuratețea Umană**: Studiile indică faptul că acuratețea umană în recunoașterea emoțiilor din imagini statice (precum cele din dataset-ul FER2013) este de aproximativ $\pm 70\%$. Această limitare provine din ambiguitatea semantică și subiectivismul etichetării.
- **Zgomotul în Etichetare (Label Noise)**: Există o probabilitate ridicată ca anumite imagini din setul de date să fie etichetate incorect sau să prezinte expresii faciale ambigue (de exemplu, confuzia frecventă între "frică" și "surpriză"), ceea ce plafonează performanța maximă teoretică a oricărui model.

Astfel, un rezultat de peste 60% este considerat competitiv, modelul demonstrând o capacitate de discriminare apropiată de nivelul uman.

V. Concluzii Finale

Acest proiect a demonstrat viabilitatea utilizării arhitecturilor complexe de tip Vision Transformer (ViT) în scenarii cu resurse limitate de date, prin aplicarea corectă a principiilor de Transfer Learning și Regularizare.

Principalele contribuții și lecții desprinse sunt:

1) **Eficiența Transfer Learning-ului**: Adaptarea unui model pre-antrenat a redus semnificativ timpul de convergență și a permis atingerea unor performanțe superioare comparativ cu antrenarea de la zero.
2) **Importanța Preprocesării și Augmentării**: Strategiile de echilibrare a claselor și augmentarea dinamică a datelor au fost critice pentru combaterea overfitting-ului într-un regim "Low-Data".
3) **Robustețea prin TTA**: Utilizarea Test-Time Augmentation a oferit un câștig incremental de performanță fără costuri suplimentare de antrenare, confirmând utilitatea abordărilor de tip "ensemble".

References

[1] L. Zhang, et al., "A Survey on Facial Expression Recognition of Static and Dynamic Emotions," *arXiv preprint arXiv:2408.03681*, 2024. https://arxiv.org/abs/2408.03681

[2] A. Dosovitskiy, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020. https://arxiv.org/abs/2010.11929

[3] H. Zhang, et al., "mixup: Beyond empirical risk minimization," *ICLR*, 2018. https://arxiv.org/abs/1710.09412

[4] X. Zhang, et al., "A Survey on Efficient Vision Transformers: Algorithms, Techniques, and Performance Benchmarking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 12, 2024. https://ieeexplore.ieee.org/document/10508493

[5] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *ICLR*, 2019. https://arxiv.org/abs/1711.05101