AVANTIX

Avancement Simulateur IQ

AVANTIX

Temps contraint : SXX

10/01/2024

Expérimentations

Paramètres fixes :

- fs = 4GSps pour 3GHz de bande utile
- NPERSEGS : 128, 256, 512, 1024, 2048, 8192, 32768, 131072
- IMG_WIDTH = 256 → fixé car quasi aucune influence sur le ratio de gain
- IMG_HEIGHT = 1024 si non précisée
- 16 842 752 échantillons (4,211ms) → Donne SXX à IMG_WIDTH colonnes pour nperseg=131075

Script : CBE/_temps_contraint/sxx/

Calcul des spectrogrammes

```
def sxx():
     # Reshape en temps
     iq = iq.unfold(0, nperseg, nperseg – noverlap)
     # Windowing par tukey (alpha = 0,25)
     window = tukey(nperseg, alpha=0,25)
     iq = iq * window[:, None]
     # Zeros padding if nfft > nperseg (avec nfft = max(IMG_HEIGHT, nperseg)
     iq = torch.cat(iq, torch.zeros(nfft-nperseg, iq.size(1))
     # FFT
     zxx = torch.fft.fft(iq)
     # FFTSHIFT
     zxx = torch.fft.fftshift(zxx) les[i*bs: (i+1)*bs])
     # SXX
     sxx = (conj(zxx)*zxx).real
     return sxx

def compute_all_sxxs(NPERSEGS):
    sxxs_lin = {}
    for nperseg in NPERSEGS:
           sxxs_lin[nperseg] = sxx(is, nperseg, noverlap, nfft)
    return sxxs_lin
```

Ratio 42 à gagner

Nouveau calcul des spectrogrammes

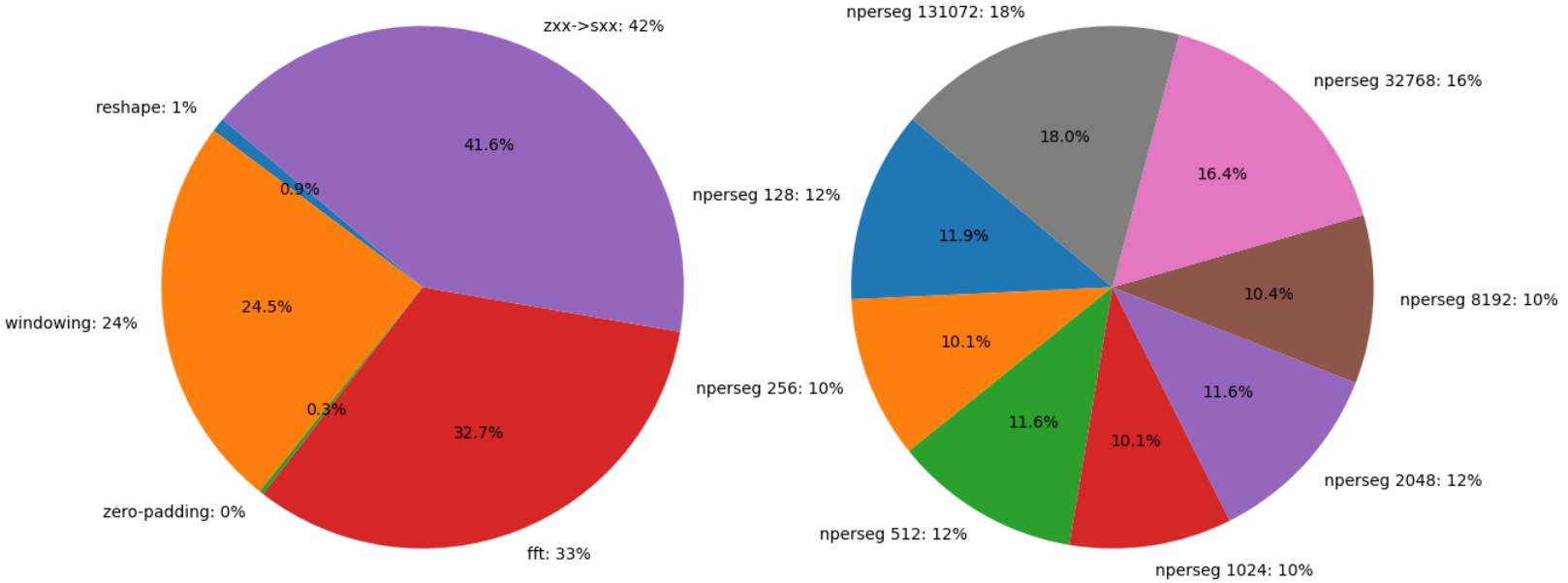
```
def sxx(iq, window):
     # Reshape en temps
     iq = iq.unfold(0, nperseg, nperseg – noverlap)
     # Windowing par tukey (alpha = 0,25)
     iq = iq * window[:, None]
     # FFT
     zxx = torch.fft.fft(iq)
     # SXX
     sxx = abs(zxx)
     return sxx

def compute_all_sxxs(NPERSEGS):
    sxxs_lin = {}
    for nperseg in NPERSEGS:
           sxxs_lin[nperseg] = sxx(iq, window)
    return sxxs_lin
```

Ratio 3.4 à gagner

Détails des durées

Total duration = 14.298 ms for 4.211 ms of iq
->overall ratio to gain: 3.4



Conclusion

Avec des gains potentiels de :

- **V100 → H100 : X3**
- **Parallélisation du calcul des spectrogrammes :**
  - **X8 : 128x256 (H, W)**
  - **X2 sinon**

Temps contraint possible sur les définitions :

- 128x256 : Ratio = 3/24
- 256x256 : Ratio = 5/6

Conséquence à diminuer la définition des images → le nombre d'images augmente et donc :

- Il ne devrait pas y avoir aucun sur le temps d'inférence mais une possible diminution des perfs de détection surtout en 256x256
- **L'étape de merge aurait plus de travail**
- **La latence sur le calcul des spectrogrammes ne nous sauvera pas si on conserve le padding**

AVANTIX

Temps contraint : Normalisation (Inférence)

10/01/2024

Etapes normalisation

```
def normalisation():
      Initialiser tensor des 2436 tuiles
      Normaliser et stocker les images :
         Mesurer d1
         Borner entre d1 et 1000*d1
         Passage en dB
         Min-Max norm entre 10*log10(q1) et 10*log10(1000*q1)
         Stocker dans la tuile
```

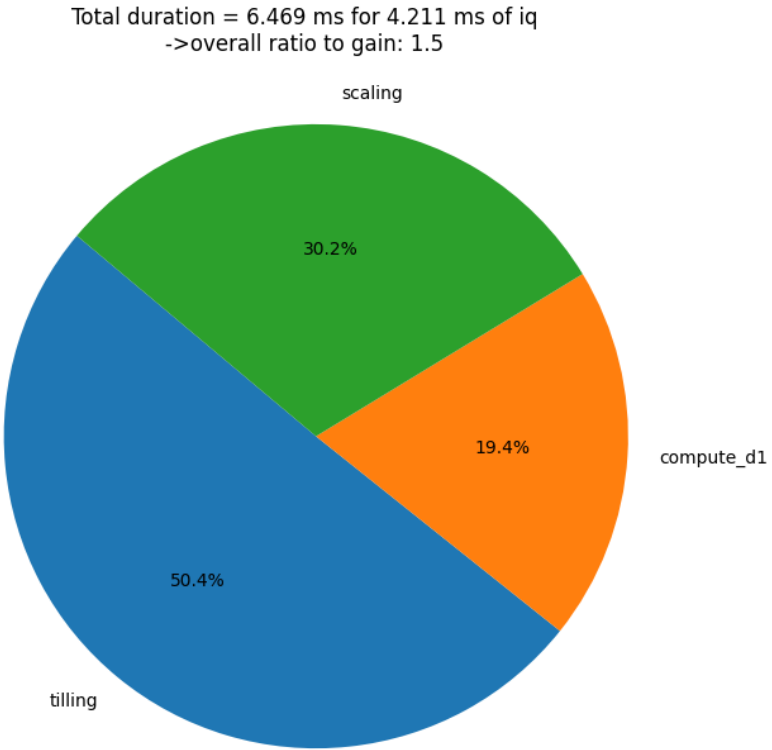
triple for loop : nperseg, height & width

Rapport 250 à gagner

Nouvelle normalisation

```
def new_normalisation():
      Mesurer d1_max sur nperseg max
      Tuiler les spectrogrammes via 1 seul for loop
      Scaler les spectrogrammes par d1_max * nperseg_max / nperseg
```

Détails des durées



AVANTIX

MERCI