**Chang\_3** (Rang 19)

*Nur insgesamt beste Abgaben Teilbereiche (alle 4 Abgaben getrennt betrachtet)*

*„Teams ranking“ „Systems ranking“*

Error Rate 0.14 (6) Chang\_1: 0.13 (8) Chang\_2 = Chang\_3 Chang\_4: 0.17 (12)

F-score 91.9 % (10) Chang\_1 = Chang\_2: 92.8 % (19) Chang\_4: 90.9 % (28)

**DOA 2.7 (1)** Chang\_4: 3.1 (2) << Chang\_1: 8.4 (22), Chang\_2: 29.7 (27)

Frame recall 90.8% (7) **Chang\_2 ≈ Chang\_4: 94.9 % (4)**  Chang\_1: 91.4 % (14)

🡪 Augmentation der Daten: Pitch-Shifting und Herstellen von *overlapped Frames* (DoA++) durch zufälliges Mixen von 2 nicht-überlappenden Audio

**Xue\_1** (Rang 5)

**Error Rate 0.06 (1)** ≈ Xue\_4: 0.07 (2) ≈ Xue\_3: 0.08 (3) ≈ Xue\_2: 0.09 (4)

**F-score 96.3 % (2) ≈ Xue\_4 ≈ Xue\_3 ≈ Xue\_2**

DOA 9.7 (13) Xue\_2: 9.2 (24) Xue\_1 ≈ Xue\_3 ≈ Xue\_4: 9.9 (28)

**Frame recall 92.3 % (3) Xue\_4 ≈ Xue\_1 ≈ Xue\_3: 92.4 % (9)**

🡪 Versionsunterschiede: Unterschiedliche Gewichtungen bei der Kombination von Log-Mel, Log-Mel&CQT und Log-Mel+BeamForming. Dabei war Gleichgewichtung (alle 1/3) am besten

🡪 Ähnlicher Ansatz wie Acoular (BeamForming, etc.)

🡪 DoA über SRP-PHAT, da Ergebnisse des Neuronalen Netzes aufgrund der über der Zeit variierenden Log-Mel-Features schwanken (dennoch als Ankerpunkt für Verknüpfung der Ergebnisse von SED<->SEL)

**Jee\_1** (Rang 10, *basiert auf Cao*)

Error Rate 0.12 (5) Jee\_2: 0.19 (14)

F-score 93.7 % (6) Jee\_2: 89.1 % (34)

**DOA 4.2 (3)** Jee\_2: 8.1 (21)

Frame recall 91.8 % (5) Jee\_2: 85.0 % (38)

🡪 Versionsunterschiede: *keine Angaben*

🡪 Bark spectrogram bringt nichts im Vergleich zu log-mel, Augmentation d. Daten mit *mixup* verbessert DoA-Error

🡪 Pooling nicht über die Zeitdomain, da sonst wichtige Infos für nächste Layer verloren gehen!

🡪 höhere Anzahl *mel bins* könnte Auflösung verbessern

**He\_2** (Rang 6)

**Error Rate 0.06 (1) = He\_1 = He\_4 (1)** He\_3: 0.08 (3)

**F-score 96.7 % (1) = He\_1 = He\_4 (1)** He\_3: 95.6% (5)

DOA 22.4 (18) ≈ He\_1 ≈ He\_3: 23.5 (37) He\_4: 26.1 (42)

**Frame recall 94.1 % (1) He\_1: 94.4 % (4)** He\_2 ≈ He\_4 ≈ He\_3: 93.5 (6)

🡪 SpecAugment funktioniert super für SED!!

🡪 Versionsunterschiede:

- He\_1 & He\_2 mit FOA, He\_3 und He\_4 mit MIC  
- 4 versch. SED-Modelle durch verschiedene augmentierte Datensätze kombiniert mit 1 DoA-Modell

**MazzonYasuda\_3**(Rang 18, *basiert auf Cao)*

**Error Rate 0.1 (3)** MY\_1, MY\_2, MY\_4 ≈ 0.13 (8)

F-score 94.2 % (5) MY\_1 = 93.3 % (16), MY\_2 = 93.0 % (18), MY\_4 = 92.0 % (22)

DOA 6.4 (8) **MY\_2: 5.0 (6)** MY\_1 ≈ MY4: 7.2 (16)

Frame recall 88.8 % (8) MY\_2 ≈ MY\_1: 88.1 % (26) MY\_4: 87.1 % (32)

🡪 Versionsunterschiede:

MY\_1:  
- 8 SELD-Modelle (blau)  
- Training der Regressionsparameter mit den 3 Training-Splits

MY\_2:  
- 8 SELD-Modelle  
- Training d. Regr.parameter mit Validation-Split

MY\_3:  
- 8 SELD-Modelle  
- keine Splits, daher nur 1 *Training dataset* & kein *ensemble via average*

MY\_4:  
- 1 SELD-Modell & keine *ensemble*-Methoden

**🡪**

**Cao\_4** (Rang 4)

**Error Rate 0.08 (2) = Cao\_1** Cao\_3: 0.1, Cao\_2: 0.12

**F-score 95.5 % (3)** Cao\_1: 95.1 % (8), Cao\_3: 94.9 % (9), Cao\_2: 93.8 % (12)

DOA 5.5 (6) = Cao\_1 = Cao\_2 (9) Cao\_3: 5.8 (12)

**Frame recall 92.2 % (4)** Cao\_1: 91.0 % (16), Cao\_3: 90.4 % (19), Cao\_2: 89.0 % (23)

🡪 Versionsunterschiede: im Paper nicht zu finden. Vielleicht nur mehrere Durchläufe?!

**🡪 Entscheidend (was viele abgeguckt haben) ist, erst SED-Netz trainieren und anschließend für DoA-Branch verwenden**

**Kapka\_2** (Rang 1)

**Error Rate 0.08 (2) = Kapka\_4 (3)** Kapka\_3: 0.1

**F-score 94.7 % (4) = Kapka\_4 (10)** Kapka\_3: 93.5 % (14)

**DOA 3.7 (2) = Kapka\_4 (3)** Kapka\_3: 4.6 (5)

**Frame recall 96.8 % (1) = Kapka\_4 (1)** Kapka\_3: 96.0 % (2)

🡪 Versionsunterschiede:

- Training von Kapka\_2 und Kapka\_4 mit allen 4 Splits der Dev-Datensätze, **Kapka\_3 mit 2, 3 und 4**

- **Classifier:** Kapka\_2: binary crossentropy loss <-> Kapka\_4: categorical crossentropy loss

- „Kapka\_1“ heißt Lewandoski\_1 und ist nur leichte Veränderung zu SELDnet

Schlussfeld:

Perezlopez „parametric Frontend“ zur DoA-Bestimmung und Filterung in Zeit & Raum  
 „deep learning Backend“ (Klassifikation) leidet unter Performanz vom Frontend

Krause deutlich weniger komplex als SELDnet, arborescentNN & depthwise separable convolutions.  
Als Inputs nur channelwise spectograms & phasegrams!

Cordourier GCC-grams, also auf wesentlichen Bereich verkleinerte GCC-PHATs  
3 parallele Branches (im fully connected-Teil): SED, Azimuth und Elevation  
12 SED-Klassen: 11 Klassen + “Garbage Class”

Chytas Kleine (100ms) vs. große (400ms) Fenstergröße -> guter frame-recall vs. gute SED  
Nur CNN ohne Recurrent (deshalb die verschiedenen Fenstergrößen?)

Anemueller group delays als Feature statt Phasenspektrum  
 Kleine Änderungen am SELDnet-System mit kaum Auswirkungen

Fazit:

SEL und SED getrennt trainieren (Sound Activity als Masking für SEL nutzen)

Datenaugmentation

SED: via SpecAugment (s. He)

SEL/DoA: mixup (s. Jee), Erweitern der 2-Quellen-Blöcke (s. Chang)

CRNN:

Input-Feature: GCC-PHAT, log-mel spectrogram

Aktivierungsfunktion: nicht **Leaky** ReLU (s. Jee)

Pooling: nicht über Zeitdomain poolen, um Auflösung hoch zu halten! (s. Jee)

Anzahl mel bins? (Nach Jee: mehr bins => höherer SED-Score möglich)

Genauigkeit von Elevation/Azimuth nicht mehr 10° sondern 1°! (Bester DoA-Error letztes Mal 2.7)