Burn Resuscitation and Management for Early Responders

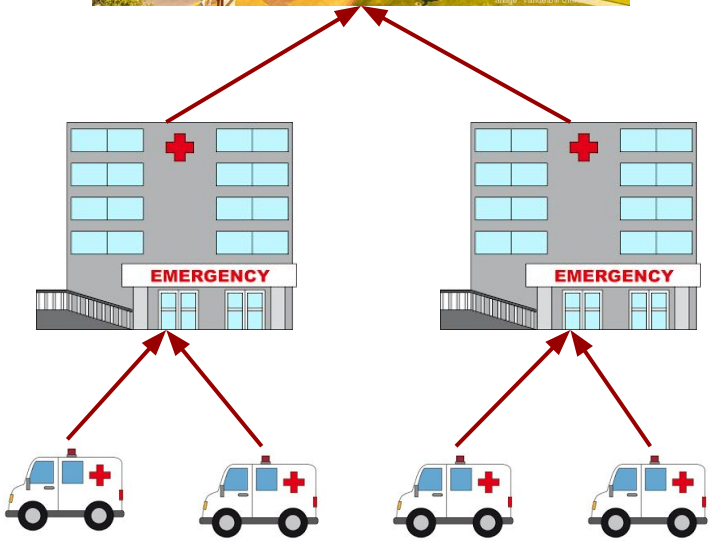
BMExtra Group:
Jacob Ayers (BME), Hannah Kang (BME), Dominique Szymkiewicz (BME), Nora Ward (BME), Thomas Yates (BME), Eric Yeats (CompE)

Contact:
Avinash Kumar M.D.

Presentation Overview

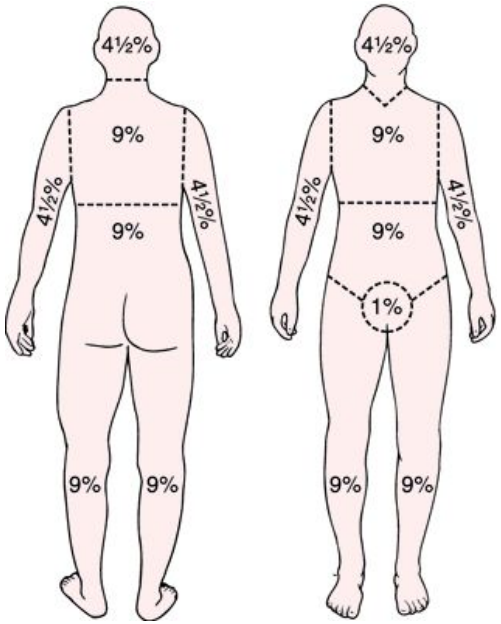
VUMC Burn ICU

- Vanderbilt Burn ICU
  - Level 1 Burn Center
  - 630 new admissions per year
    - Majority transferred from E.R.
  - Primary Contact: Dr Avinash Kumar

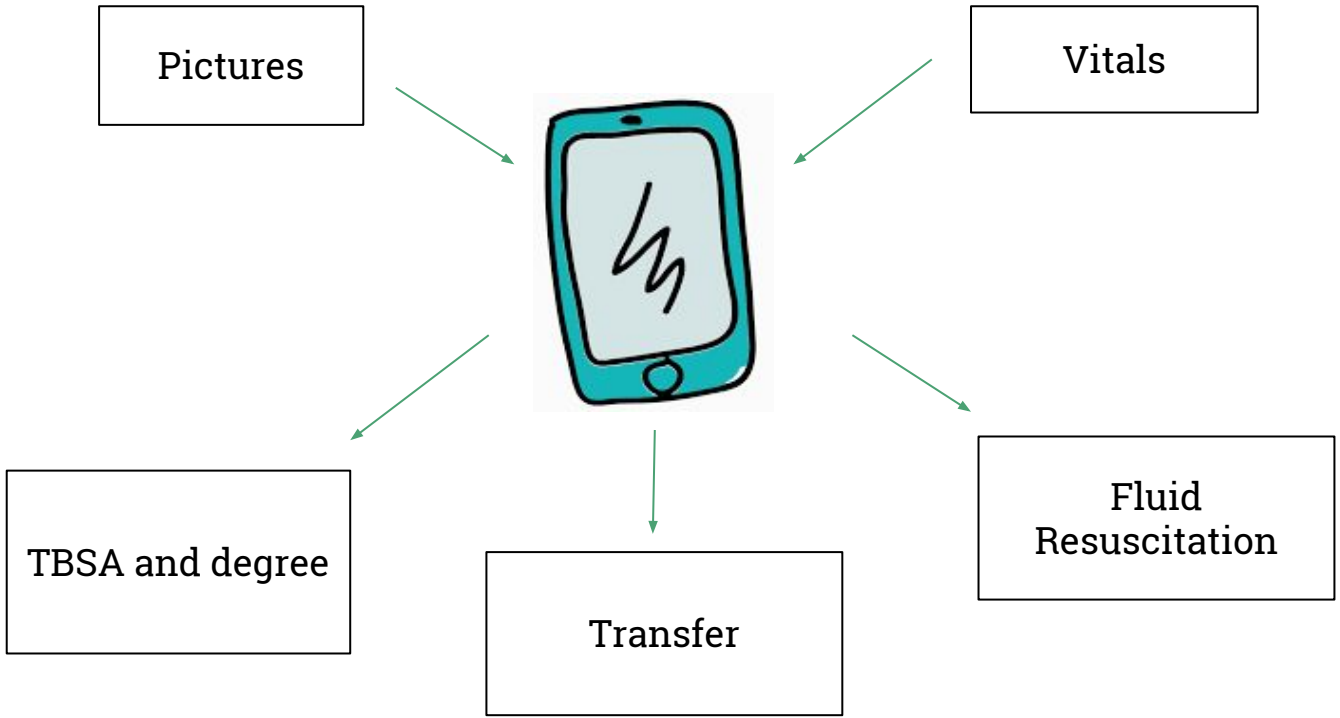


Problem Statement

- Current System: Wallace Rules of Nine
  - Adjust for age and BMI
- Problems:
  - Overestimation of burn percentage
  - Overburden Burn centers with patients
- Goal: Develop system to rapidly and accurately determine TBSA



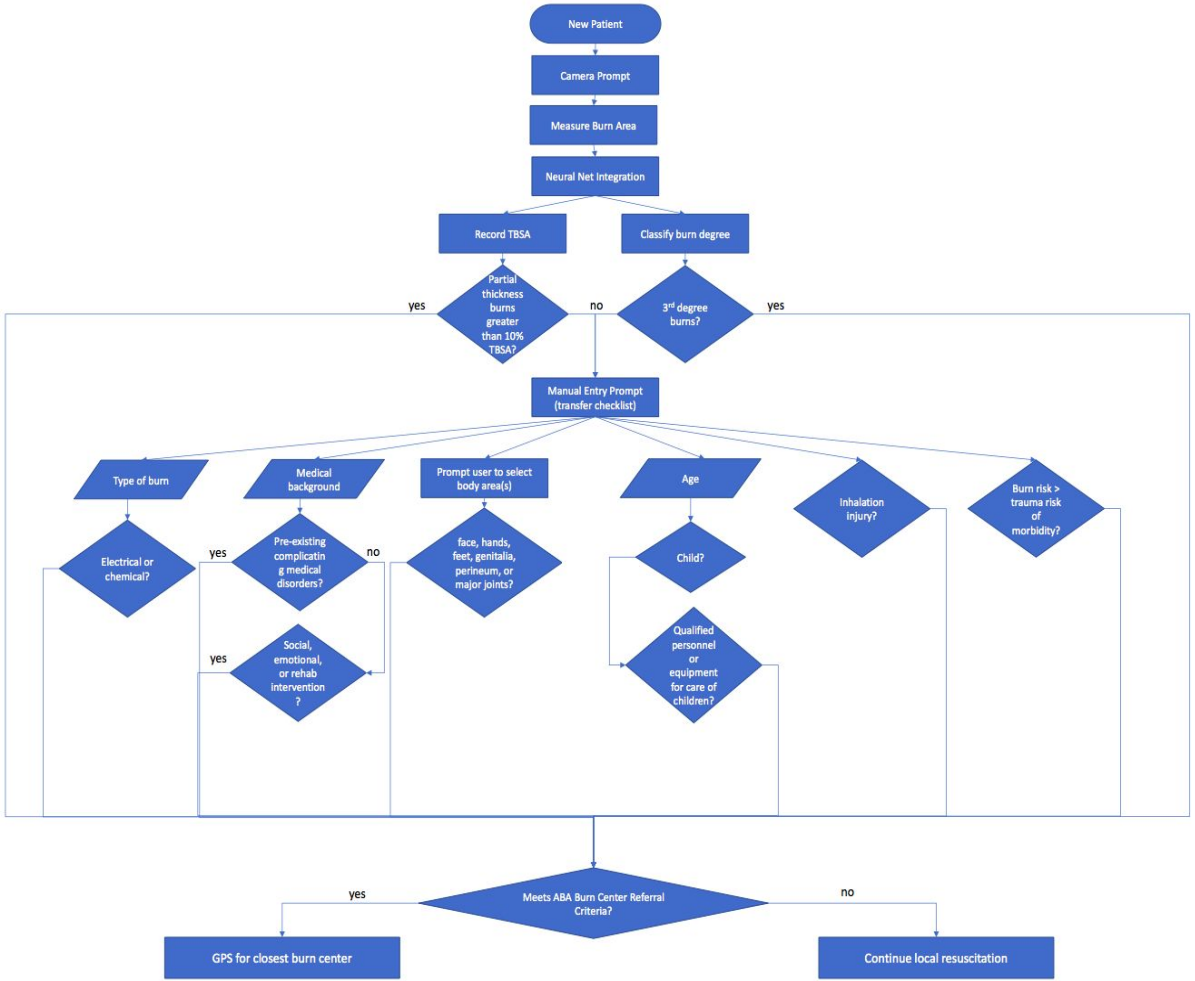
Our Solution



Needs Assessment

1. Infrastructure compatibility
2. Safety
3. Patient Efficacy
4. Performance Capabilities
5. Cost Efficacy

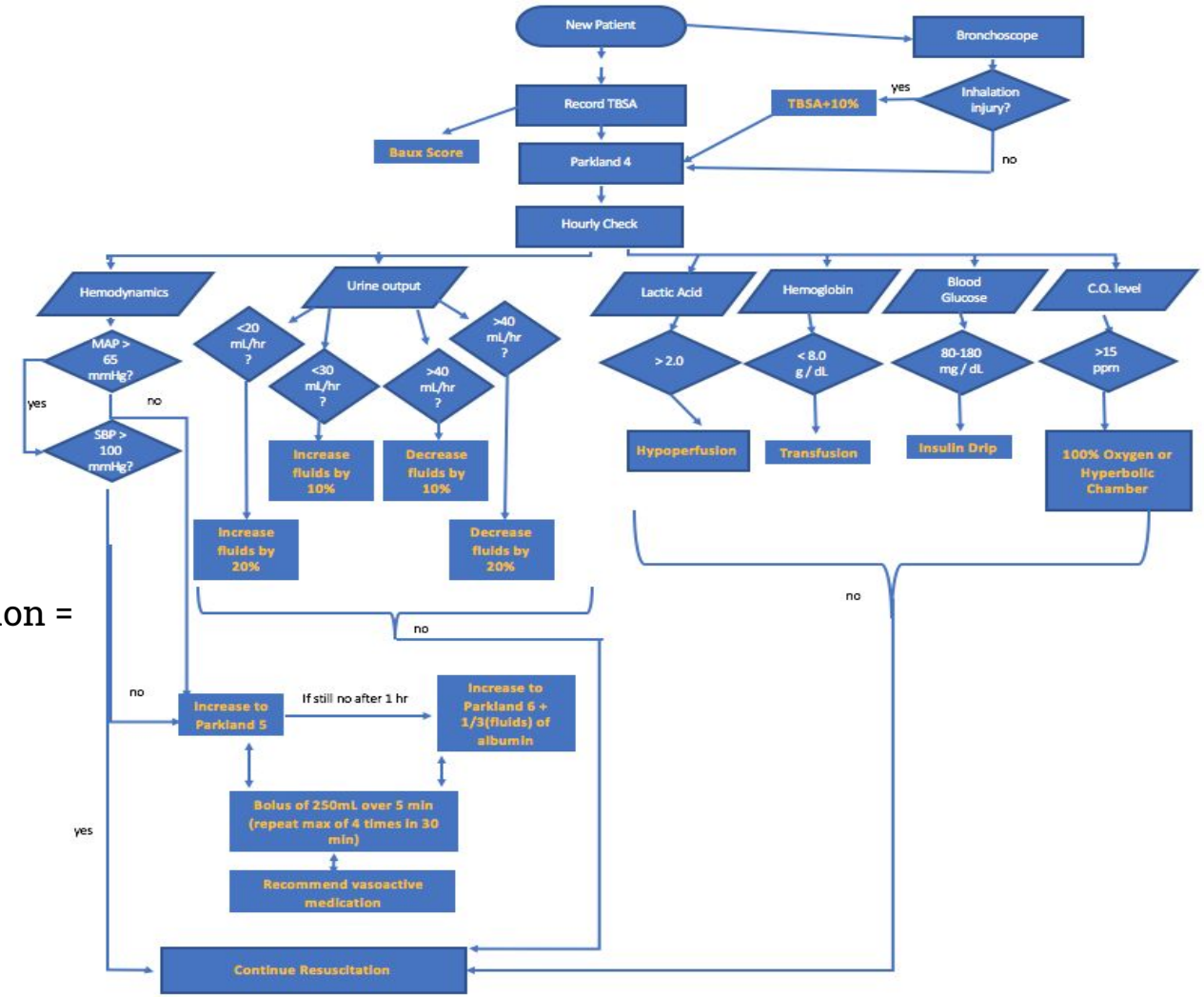
Review of Flow Charts: Transfer



Review of Flow Charts: Resuscitation

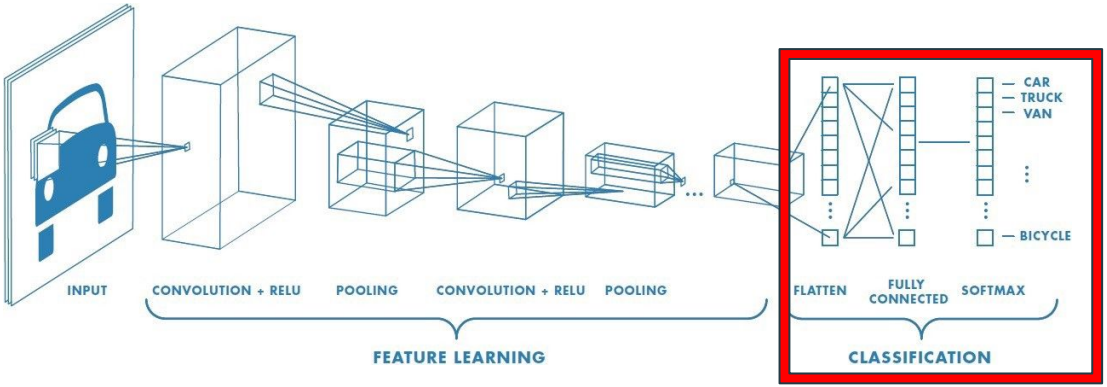
Parkland 4 Formula

Volume of Lactated Ringers Solution = (4mL) * (TBSA %) * (weight in kg)



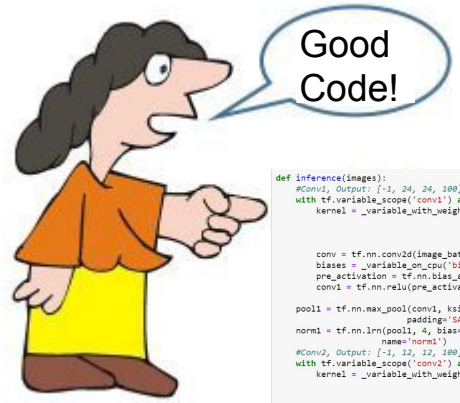
Neural Network Update

- Convolutional Neural Network
  - 50x50x3 Images ➜ Softmax Output



Neural Network Update

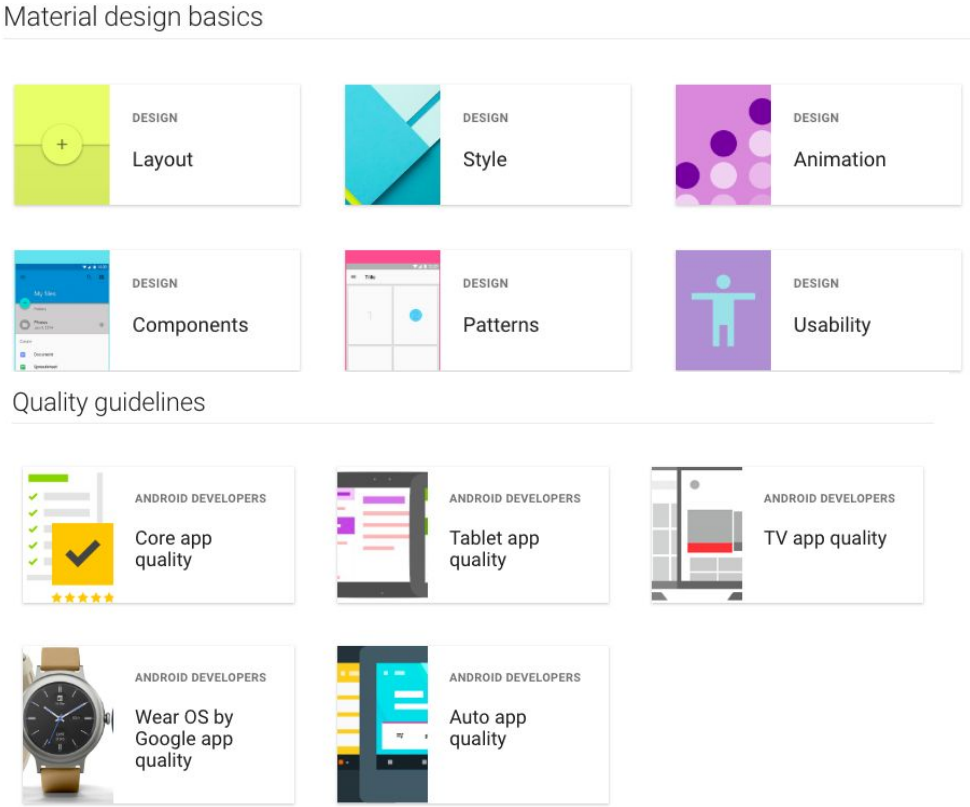
- ConvNet Model Is Built
  - 3 Conv/Relu/Pool Layers
    - 50x50x3 → 6x6x100
  - 2 Dense Layers
    - 1024 Nodes
  - 1 Softmax Layer
    - 3 Possible Outputs
- Working on reading images from directory
- Learning to train the ConvNet
  - Tensorflow specific language and functions




```
def inference(images):
    #Conv1, Output: [-1, 24, 24, 100]
    with tf.variable_scope('conv1') as scope:
        kernel = _variable_with_weight_decay('weights',
                                             shape=[50,50,3,100],
                                             stddev=5e-2,
                                             wd=None)
        conv = tf.nn.conv2d(image_batch, kernel, [1,1,1,1], padding='VALID'
        biases = _variable_on_cpu('biases', [100], tf.constant_initializer(
        pre_activation = tf.nn.bias_add(conv, biases)
        conv1 = tf.nn.relu(pre_activation, name=scope.name)

    pool1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,1,1,1],
                           padding='SAME', name='pool1')
    norm1 = tf.nn.lrn(pool1, 4, bias=1.0, alpha=0.001 / 9.0, beta=0.75,
                      name='norm1')
    #Conv2, Output: [-1, 12, 12, 100]
    with tf.variable_scope('conv2') as scope:
        kernel = _variable_with_weight_decay('weights',
                                             shape=[5,5,100,100],
                                             stddev=5e-2,
                                             wd=None)
        conv = tf.nn.conv2d(norm1, kernel, [1,1,1,1], padding='SAME')
        biases = _variable_on_cpu('biases', [100], tf.constant_initializer(
        pre_activation = tf.nn.bias_add(conv, biases)
        conv2 = tf.nn.relu(pre_activation, name=scope.name)

    pool2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1],
                           strides=[1,2,2,1], padding='SAME', name='pool2')

    norm2 = tf.nn.lrn(pool2, 4, bias=1.0, alpha=0.001 / 9.0, beta=0.75,
                      name='norm2')

    #Conv3, Output: [-1, 6, 6, 100]
    with tf.variable_scope('conv3') as scope:
        kernel = _variable_with_weight_decay('weights',
                                             shape=[5,5,100,100],
                                             stddev=5e-2,
                                             wd=None)
        conv = tf.nn.conv2d(norm2, kernel, [1,1,1,1], padding='SAME')
        biases = _variable_on_cpu('biases', [100], tf.constant_initializer(
        pre_activation = tf.nn.bias_add(conv, biases)
        conv3 = tf.nn.relu(pre_activation, name=scope.name)

    norm3 = tf.nn.lrn(conv3, 4, bias=1.0, alpha=0.001 / 9.0, beta=0.75,
                      name='norm3')
```

User Interface Needs (Functionality)

- Android Developers
  - Material Design Guidelines
    - Visual/Navigation patterns
  - App Quality Guidelines
    - Compatibility
    - Performance
    - Security



User Interface Needs (Clinician)

- User interface should have color coded responses
  - Indicate proximity to threshold
  - Visual alert if patient vitals are dangerous

- Clinicians would prefer to have a visual chart within the app
  - More clear vitals trends
  - Prevents overlooking changes in vitals

Wireframe Mockup Needs

Requirements:

- Integrate into Android and React Native
- User friendly for our team
- Options for user authentication to be built in

Potential Vendors:

- WireFramePro - $39 per month (3 team members)
- BuilderX - $29.50 per month per user (~ 3 users for our team)
- Balsamiq - $89 per user

IRB Requirements

- Prototype Testing
  - Need approval from clinicians to collect data
  - Access to clinician notes about patient recommendations
  - Research Studio
- Goals:
  - Further train diagnostic portion of application
  - Assess accuracy of TBSA measurements
    - Compare our treatment recommendations to Physicians
  - Understand additional needs for the application

Next Steps

1. Acquire wireframe software and begin UI prototype
2. Continue image labeling and burn differentiation
3. Digitize flow charts within UI
4. IRB initiation